Admissible Strategies in Timed Games

Nicolas Basset, Jean-François Raskin, Ocan Sankur

Admissible Strategies in Timed Games

Nicolas Basset[1], Jean-François Raskin[1], and Ocan Sankur[2]

[1] Université libre de Bruxelles, Brussels, Belgium
{nbasset, jraskin}@ulb.ac.be
[2] CNRS, IRISA, Rennes, France
ocan.sankur@irisa.fr

Abstract. In this paper, we study the notion of admissibility in timed games. First, we show that admissible strategies may not exist in timed games with a continuous semantics of time, even for safety objectives. Second, we show that the discrete time semantics of timed games is better behaved w.r.t. admissibility: the existence of admissible strategies is guaranteed in that semantics. Third, we provide symbolic algorithms to solve the model-checking problem under admissibility and the assume-admissible synthesis problem for real-time non-zero sum n-player games for safety objectives.

1 Introduction

An embedded controller is a reactive system that maintains a continuous interaction with its environment and has the objective to enforce outcomes, from this interaction, that satisfy some good properties. As the actions taken by the environment in this interaction are out of the direct control of the controller, those actions should be considered as adversarial. Indeed, a controller should be correct no matter how the environment in which it operates behaves. As reactive systems most often exhibit characteristics, like real-time constraints, concurrency, or parallelism, etc., which make them difficult to develop correctly, formal techniques have been advocated to help to their systematic design. One well-studied formal technique is model checking [3] which compares a model of a system with its specification. Model-checking either provides a proof of correctness of the model of the controller within its environment or provides a counter-example that can be used to improve the design.

A scientifically more challenging technique is *synthesis* that uses algorithms that transform the specification of a reactive system and a model of its environment into a correct system, i.e., a system that enforces the specification no matter how the environment behaves. Synthesis can take different forms: from computing optimal values of parameters to the full-blown automatic synthesis of a model of the system's components. Albeit this diversity, one mathematical model has emerged to perform synthesis for reactive systems: two-player zero-sum games played on graphs; and the main solution concept for those games is the notion of winning strategy. Zero-sum timed games played on timed automata (defined by [1]) have been introduced in [27] as a formal model for the synthesis of reactive systems with *timed specifications*. A practical algorithm for the problem was first presented in [17] and implemented in the tool Uppaal-Tiga [5].

Timed games, as defined in [27] and in almost all subsequent works, see e.g. [2, 17, 15, 16], are zero-sum games. In zero-sum games, the environment is considered as *fully antagonist*. The zero-sum game abstraction is often used because it is *simple* and *sound*: a winning strategy against an antagonistic environment is winning against any environment including obviously those that

This work was partially supported by the ERC Starting grant 279499 (inVEST), the ARC project "Non-Zero Sum Game Graphs: Applications to Reactive Synthesis and Beyond" (Fédération Wallonie-Bruxelles, J.-F. Raskin is Professeur Francqui de Recherche.

strive to secure their own objective. But, in general the zero-sum hypothesis is a bold abstraction of reality: most often the environment has its own objective which, in general, does not correspond to that of falsifying the specification of the controller. Then, it should be clear that the zero-sum approach may fail to find a winning strategy even if solutions exist when the objective of the environment is taken into account, or it may produce sub-optimal solutions because those solutions are overcautious in order to be able to face with all possible behaviors of the environment, even if they are in contradiction with the environment's objectives. Recently, several new solution concepts for synthesis of reactive systems that take the objectives of the environment into account, and so relax the fully adversarial assumption, have been introduced [10]. One approach that is particularly promising is based on the notion of admissible strategies [7, 23, 13, 12, 11].

Assume Admissible Synthesis In [12], we have introduced a new synthesis rule based on *admissibility* in the general case of n-player multiplayer games. This synthesis rule can be summarized as follows. For a player with objective ϕ, a strategy σ is dominated by σ' if σ' does as well as σ w.r.t. ϕ against all strategies of the other players, and better for some of those strategies. A strategy σ is *admissible* if it is not dominated by another strategy. Starting from the fact that only admissible strategies should be played by rational players (dominated strategies being clearly sub-optimal options), when synthesizing a controller, we search for an admissible strategy that is winning *against all admissible strategies* of the environment. Assume admissible synthesis is sound: if all players choose admissible strategies that are winning against all admissible strategies of the other players, the objectives of all players is guaranteed to be satisfied.

Assume Admissible Timed Synthesis In the classical setting of game graphs with ω-regular objectives, admissibility is well behaved: admissible strategies always exist in perfect information n-player game graphs with ω-regular objectives, both for turn-based games [7, 23, 13] and for concurrent games [4]. By contrast, in this paper, we show that, in the *continuous time* semantics, players in a timed game are not guaranteed to have admissible strategies. This is because in some timed games there may not exist an optimal time to play. This is the case for example if a player has to play as soon as possible but strictly after a given deadline. We exhibit concrete games with this property. We also show that those problems are an artefact of the continuous time semantics. In contrast, in the discrete-time semantics of timed games, admissible strategies always exist.

To obtain our results in the discrete-time semantics we provide a reduction to finite concurrent games with an additional player that arbitrates situations in which several players propose to play at the exact same time. While the reduction to finite concurrent games is adequate to obtain theoretical results, it is not practical. This is why we define symbolic algorithms based on zones to solve the model-checking under admissible strategies and the assume admissible synthesis problem for safety objectives. To obtain those symbolic algorithms, we show how to use (continuous) timed zones to represent efficiently sets of discrete time valuations. We believe that those results are also interesting on their own. Note that it is possible to solve discrete-time games by enumerative techniques [25]; however, our algorithms require representing complex sets of states, so being able to solve a given game is not sufficient, and we do need some form of succinct representation.

Other related works Related works on zero-sum timed games have been given above. To the best of our knowledge, our work is the first to deal with admissibility for timed games. In this paragraph we discuss several works related to admissibility in (untimed) games.

Other works in the literature propose the use of Nash equilibria (NE) in n-players non-zero sum games to model variants of the reactive synthesis problem. Most notably, assume-guarantee

synthesis, based on secure equilibria [19] (refining Nash equilibria), has been proposed in [18], while cooperative rational synthesis has been proposed in [24], and non-cooperative rational synthesis in [26]. In the context of infinite duration games played on graphs, one well known limitation of Nash equilibria is the existence of non-credible threats. Refinements of the notion of NE, like sub-game perfect equilibria (SPE), have been proposed to overcome this limitation. SPE for games played on graphs have been studied in e.g. [29, 14]. Admissibility does not suffer from this limitation. In [23], Faella proposes several alternatives to the notion of winning strategy including the notion of admissible strategy. His work is for two-players but only the objective of one player is taken into account, the objective of the other player is left unspecified. In that work, the notion of admissibility is used to define a notion of best-effort in synthesis. The notion of admissible strategy is definable in strategy logics [20, 28] and decision problems related to the assume-admissible rule can be reduced to satisfiability queries in such logics. This reduction does not lead to worst-case optimal algorithms; we presented worst-case optimal algorithms in [21] based on our previous work [13].

The only works that we are aware of and that consider non-zero sum timed games are the following two papers [8, 9] that study decision problems related to the concept of Nash equilibria and not to the concept of admissibility.

2 Admissibility in Concurrent Games

Let $P = \{1, 2, \dots n\}$ denote a set of players. A *concurrent game* played by players P is a tuple $\mathcal{G} = (S, s_{\mathsf{init}}, \Sigma, (\mathsf{M}_i)_{i \in P}, \delta)$ where,

- S is a set of *states*, and $s_{\mathsf{init}} \in S$ the *initial state*;
- Σ is a set of *moves*;
- For all $i \in P$, $\mathsf{M}_i : S \to 2^{\Sigma} \setminus \{\emptyset\}$ assigns to every state $s \in S$ and player i the set of *available moves* from state s.
- $\delta : S \times \Sigma \times \ldots \times \Sigma \to S$ is the *transition function.*

The game is called *finite* if S and Σ are finite. We write $\mathsf{M}(s) = \mathsf{M}_1(s) \times \ldots \times \mathsf{M}_n(s)$ for every $s \in S$. A *history* is a finite path $h = s_1 s_2 \dots s_N \in S^*$ such that (i) $N \in \mathbb{N}$; (ii) $s_1 = s_{\mathsf{init}}$; and (iii) for every $2 \leq k \leq N$, there exists $(a_1, \dots, a_n) \in \mathsf{M}(s_{k-1})$ with $s_k = \delta(s_{k-1}, a_1, \dots, a_n)$. A *run* is defined similarly as a history except that its length is infinite. For a history or a run ρ, let us denote its i-th state by ρ_i. The game is played from the initial state s_{init} for an infinite number of rounds, producing a run. At each round $k \geq 1$, with current state s_k, all players i select simultaneously moves $a_i \in \mathsf{M}_i(s_k)$, and the state $\delta(s_k, a_1, \dots, a_n)$ is appended to the current history.

It is often convenient to consider a player i separately and see the set of other players $P \setminus \{i\}$ as a single player denoted $-i$. Hence, the set of moves of $-i$ in state s is $\mathsf{M}_{-i}(s) = \prod_{j \in P \setminus \{i\}} \mathsf{M}_j(s)$.

An *objective* ϕ is a subset of runs of the game. We assume that concurrent games are equipped with a function Φ mapping all players $i \in P$ to an objective $\Phi(i)$. Thus, a run ρ is winning for player i iff $\rho \in \Phi(i)$. An objective $\phi \subseteq S^{\omega}$ is a *simple safety* objective if there exists $B \subseteq S$ such that $\rho \in \phi$ if, and only if $\forall j, \rho_j \notin B$; and for all $s \in B$ and $m \in \mathsf{M}(s)$, $\delta(s, m) \in B$. In other terms, once B is reached, the play never leaves B. The set B is informally called *bad states* for the objective ϕ. Note that contrary to general safety objectives, simple safety objectives are prefix independent. Also, any safety objective can be turned into a simple safety objective by modifying the underlying concurrent game. Games equipped with simple safety objectives are called simple safety games.

A strategy for player i is a function σ from histories to moves of player i such that for all histories h: $\sigma(h) \in \mathsf{M}_i(s)$ where s is the last state of h. We denote by $\Gamma_i(\mathcal{G})$ the set of player i's

strategies in the game; we might omit $\mathcal{G}$ if it is clear from context. A *strategy profile* $\boldsymbol{\sigma}$ for a subset $A \subseteq P$ of players is a tuple $(\boldsymbol{\sigma}_i)_{i\in A}$ with $\boldsymbol{\sigma}_i \in \Gamma_i$ for all $i \in A$. When the set of players A is omitted, we assume $A = P$. Let $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_i)_{i\in P}$ be a strategy profile. Then, for all players i, we let $\boldsymbol{\sigma}_{-i}$ denote the restriction of $\boldsymbol{\sigma}$ to $P \setminus \{i\}$ (hence, $\boldsymbol{\sigma}_{-i}$ can be regarded as a strategy of player $-i$ that returns, for all histories h, a move from $\mathsf{M}_{-i}(s)$ where s is the last state of h). We denote by Γ_{-i} the set $\{\boldsymbol{\sigma}_{-i} \mid \boldsymbol{\sigma} \in \Gamma\}$. We sometimes denote by $\boldsymbol{\sigma}$ the pair $(\boldsymbol{\sigma}_i, \boldsymbol{\sigma}_{-i})$. For any history h, let $\boldsymbol{\sigma}(h) = (\boldsymbol{\sigma}_i(h))_{i\in A}$ and be the tuple of choices made by all players (when they play from h according to $\boldsymbol{\sigma}$) and the resulting state, respectively. We let $\mathtt{Out}(\boldsymbol{\sigma})$ be the *outcome of* $\boldsymbol{\sigma}$, i.e. the unique run $\rho = s_1 s_2 \cdots$ such that $s_k = \delta(s_{k-1}, \boldsymbol{\sigma}(s_1 \cdots s_{k-1}))$.

Assume the game we consider has winning condition Φ. Then, we say that $\boldsymbol{\sigma}$ *is winning for* i*, from* h, written $\boldsymbol{\sigma} \models_h \Phi(i)$, if h is a prefix of $\mathtt{Out}(\boldsymbol{\sigma})$ and $\mathtt{Out}(\boldsymbol{\sigma}) \in \Phi(i)$. We write $\sigma \models_h \Phi(i)$, if for every $\tau \in \Gamma_{-i}$ such that h is a prefix of $\mathtt{Out}((\sigma, \tau))$ it holds that $\mathtt{Out}((\sigma, \tau)) \in \Phi(i)$.

Dominance and admissibility Fix a game $\mathcal{G}$ and a player i. Given two strategies $\sigma, \sigma' \in \Gamma_i$, we say that σ is *weakly dominated* by σ', denoted $\sigma \preccurlyeq \sigma'$ if for all $\sigma_{-i} \in \Gamma_{-i}$, $(\sigma, \sigma_{-i}) \models \Phi(i)$ implies $(\sigma', \sigma_{-i}) \models \Phi(i)$. Intuitively, this means that σ' is *not worse* than σ, because it yields a winning outcome (for i) every time σ does. When $\sigma \preccurlyeq \sigma'$ but $\sigma' \not\preccurlyeq \sigma$ we say that σ is *dominated* by σ'. Note that $\sigma \prec \sigma'$ if and only if $\sigma \preccurlyeq \sigma'$ and there exists at least one $\sigma_{-i} \in \Gamma_{-i}$, such that $(\sigma, \sigma_{-i}) \not\models \Phi(i)$ and $(\sigma', \sigma_{-i}) \models \Phi(i)$. That is, σ' is now *strictly better* than σ because it yields a winning outcome for i every time σ does; but i secures a winning outcome against at least one strategy of the other players by playing σ' instead of σ. A strategy is called *admissible* if it is not dominated.

Theorem 1 ([4]). *For every finite concurrent game, for all objectives, the set of admissible strategies of each player is non-empty.*

Now that we have defined a notion of dominance on *strategies*, let us turn our attention to a more local definition of dominance on *moves*. Let h be a history. We say that a move $a \in \mathsf{M}_i$ is *h-dominated* by another move $a' \in \mathsf{M}_i$ iff for all $\sigma \in \Gamma_i$ s.t. $\sigma(h) = a$, there exists $\sigma' \in \Gamma_i$ s.t. $\sigma'(h) = a'$ and $\sigma \prec_h \sigma'$. We denote this by $a <_h a'$. If a move a is not h-dominated by any move, we say that a is *h-admissible*. This allows us to define a more local notion of dominated strategy: a strategy σ of player i is called *locally-admissible* (LA for short) if for every h, $\sigma(h)$ is an h-admissible move. By definition, all admissible strategies are also LA, but the converse only holds for simple safety games.

Theorem 2 ([4]). *In concurrent finite simple safety games, a strategy is locally admissible if, and only if it is admissible.*

We close these preliminaries by explaining how to associate *values* to histories and moves. First, the value of history h for player i is defined as follows. $\chi^i_h = 1$ if $\exists \sigma \in \Gamma_i \ \forall \sigma_{-i} \in \Gamma_{-i}, (\sigma_i, \sigma_{-i}) \models_h \Phi(i)$; $\chi^i_h = -1$ if $\forall \boldsymbol{\sigma} \in \Gamma, \boldsymbol{\sigma} \not\models_h \Phi(i)$; and $\chi^i_h = 0$ otherwise.

So the intuition is that: (i) $\chi^i_h = 1$ iff i has a winning strategy from h; (ii) $\chi^i_h = -1$ iff *no outcome* is winning for i from h; and (iii) $\chi^i_h = 0$ when i has no winning strategy from h but can still win with the help of other players. Thus, $\chi^i_h = -1$ is stronger than saying that i has no winning strategy from h, since, in this case, i can never win, even with the help of other players. When the other players can help, we have rather $\chi^i_h = 0$, which means that there is some strategy σ of i such that there is a profile $\boldsymbol{\sigma}$ with $\boldsymbol{\sigma}_i = \sigma$ and $\boldsymbol{\sigma} \models_h \Phi(i)$.

Lemma 1 ([4]). *In finite concurrent games, for any player i, history h that ends in a state s, and moves $a, b \in \mathsf{M}_i(s)$, we have $a <_h b$ if, and only if the conjunction of the following conditions holds:*

(i) $\chi^i_{h\delta(s,a,c)} \leq \chi^i_{h\delta(s,b,c)}$ *for every* $c \in \mathsf{M}_{-i}(s)$*;*
(ii) $\chi^i_{h\delta(s,a,c)} < \chi^i_{h\delta(s,b,c)}$ *for at least one* $c \in \mathsf{M}_{-i}(s)$*;*
(iii) if $\chi^i_{h\delta(s,a,c)} = \chi^i_{h\delta(s,b,c)} = 0$ *then* $\delta(s,a,c) = \delta(s,b,c)$*, for every* $c \in \mathsf{M}_{-i}(s)$*.*

3 Multi-Player Timed Games

In this section, we define multiplayer timed games and apply previously defined admissibility notions to this setting.

Given a finite set of clocks X, we call the elements of $\mathbb{R}^X_{\geq 0}$ *valuations*, and those of $\mathbb{N}^X$ *discrete valuations*. Let $\mathbb{N}^X_{\leq M}$ denote the subset of $\mathbb{N}^X$ in which all components are bounded by M. For a subset $R \subseteq X$ and a valuation ν, $\nu[R \leftarrow 0]$ is the valuation defined by $\nu[R \leftarrow 0](x) = \nu(x)$ for $x \in X \setminus R$ and $\nu[R \leftarrow 0](x) = 0$ for $x \in R$. Given $d \in \mathbb{R}_{\geq 0}$ and a valuation ν, the valuation $\nu + d$ is defined by $(\nu + d)(x) = \nu(x) + d$ for all $x \in X$. We extend these operations to sets of valuations in the obvious way. We write $\mathbf{0}$ for the valuation that assigns 0 to every clock.

An *atomic clock constraint over* X is a formula of the form $k \leq x \leq l$ or $k \leq x - y \leq l$ where $x, y \in X$, $k, l \in \mathbb{Z} \cup \{-\infty, \infty\}$. A *guard* is a conjunction of atomic clock constraints. A valuation ν satisfies a guard g, denoted $\nu \models g$, if all constraints are satisfied when each $x \in X$ is replaced with $\nu(x)$. We write Φ_X for the set of guards built on X.

Let P be a finite a set of players. A *multi-player timed game* between players P is a tuple $\mathcal{G} = (\mathcal{L}, \iota, \mathcal{I}, X, (\Delta_i)_{i \in P})$ where (i) $\mathcal{L}$ is a finite set of *locations*, (ii) ι is the *initial location*, (iii) X is a finite set of *clocks*, (iv) $\mathcal{I} : \mathcal{L} \to \Phi(X)$ is the *invariant* associated to each location; we assume that invariants only contain upper bounds on clocks, (v) $\Delta_i \subseteq \mathcal{L} \times \Phi(X) \times 2^X \times \mathcal{L}$, the set of *Player-i edges*: in each tuple (ℓ, g, R, ℓ'), ℓ is the *source location*, g is the *guard*, R the *reset set*, and ℓ' the *target location*. For any edge $e \in \Delta_i$, let us denote by $(\ell_e, g_e, R_e, \ell'_e)$ the tuple associated to it.

The Discrete-Time Semantics In this paper, timed games are equipped with a discrete time semantics described now. We explain later why problems happen when a continuous time semantics is considered instead.

In the *discrete-time* semantics, not only are all delays restricted to be discrete, but we also assume that each clock tick is globally observable by all players. Thus, at each clock tick, all players simultaneously decide either to wait another clock tick, or to take an enabled edge. The non-determinism between suggested edges is resolved by an additional player called *scheduler*.

Given state (ℓ, ν) and an edge $e = (\ell, g, R, \ell')$ such that $\nu \models g$, and $\nu[R \leftarrow 0] \models \mathcal{I}(\ell')$, let us write $(\ell', \nu') = \mathtt{Succ}_e((\ell, \nu))$ where $\nu' = \nu[R \leftarrow 0]$.

Consider a bound $M > 0$ larger than all constants that appear in the guards and define the operation $+_M$ by $a +_M b = \min(M, a + b)$ for every $a, b \in \mathbb{R}$. We define the semantics of a timed game $\mathcal{G} = (\mathcal{L}, \iota, \mathcal{I}, X, (\Delta_i)_{i \in P})$ as a concurrent game $\mathcal{D}_M(\mathcal{G}) = (S, s_{\mathsf{init}}, \Sigma, (\mathsf{M}_i)_{i \in P'}, \delta)$ where $P' = P \cup \{\mathtt{sched}\}$. Let $\mathfrak{S}(n)$ denote the set of permutations over $\{1, 2, \ldots, n\}$. We have $S = \{(\ell, \nu) \in \mathcal{L} \times \mathbb{N}^X_{\leq M} \mid \nu \models \mathcal{I}(\ell)\}$, $\Sigma = \cup_{i \in P} \Delta_i \cup \{\bot\} \cup \mathfrak{S}(n)$ where $\bot$ is a fresh symbol. For every $(\ell, \nu) \in S$, and $i \in P$, we have $\mathsf{M}_i(\ell, \nu) = \{e \in \Delta_i \mid \nu \models g_e \wedge \mathcal{I}(\ell), \nu[R_e \leftarrow 0] \models \mathcal{I}(\ell'_e)\} \cup \{\bot \mid \nu +_M 1 \models \mathcal{I}(\ell)\}$. For player $\mathtt{sched}$, we have $\mathsf{M}_{\mathtt{sched}}(\ell, \nu) = \mathfrak{S}(|P|)$. Note that $\mathcal{D}_M(\mathcal{G})$ is a finite concurrent game due to the bound M.

The transition function δ is defined from the current state (ℓ, ν) given moves $m_1, \ldots, m_n$ chosen by the players of P and a permutation π chosen by the scheduler as follows:

$$\delta((\ell, \nu), m_1, \ldots, m_n, \pi) = \begin{cases} (\ell, \nu +_M 1) & \text{if } \forall i \in P, m_i = \bot, \\ (\ell', \nu') & \text{if } i = \arg\min_{j \in P : m_j \in \Delta_j} \pi(j), \\ & m_i = (\ell, g, R, \ell'), \nu' = \nu[R \leftarrow 0]. \end{cases}$$

The intuition of the game is that at each discrete time step, each player can choose either to wait, or to switch state by picking an edge. If several players pick edges, then the player `sched` determines, by the permutation it has chosen, which edge is to be taken. In general, one add fairness constraints for the scheduler by specifying an objective for this player. However, we consider safety objectives in the present work, for which fairness is not useful.

In the rest of the paper, we only consider timed games with non-strict guards, since any strict constraint can be converted into a non-strict one when working in discrete time.

We denote by $\Delta_i(s) = \mathsf{M}_i(s)\backslash\{\bot\}$ the set of edges of player i available in s and by $\Delta_{-i}(s) = \cup_{j\in P\backslash\{i\}}\mathsf{M}_j(s)\backslash\{\bot\}$ the other edges available in s.

Non-Existence of Admissible Strategy in Continuous-Time Semantics We now show that admissible strategies are not guaranteed to exist if one considers a continuous-time semantics instead of the discrete time semantics.

In the *continuous-time* semantics, all players simultaneously suggest moves that are pairs of delay and edges to be taken, and a move with the least delay is taken. The precise choice of the edge with the least delay is determined by an additional player, named *scheduler*, which determines a priority order between players.

Given a timed game $\mathcal{G} = (\mathcal{L}, \iota, \mathcal{I}, X, (\Delta_i)_{i\in P})$ we define an infinite-state concurrent game $\mathcal{C}(\mathcal{G}) = (S^c, \Sigma^c, s^c_{\mathsf{init}}, (\mathsf{M}^c_i)_{i\in P'}, \delta^c)$ where $P' = P \cup \{\mathtt{sched}\}$. We have $S^c = \{(\ell,\nu) \in \mathcal{L}\times\mathbb{R}^X_{\geq 0} \mid \nu \models \mathcal{I}(\ell)\}$. The moves are $\Sigma^c = \{(d,e) \mid d \in \mathbb{R}_{\geq 0}, e \in \cup_i\Delta_i\} \cup \mathfrak{S}(|P|)$, and $\mathsf{M}^c_i((\ell,\nu)) = \{(d,e) \mid d \geq 0, e \in \Delta_i, \nu + d \models g_e \cap \mathcal{I}(\ell) \wedge \nu[R_e \leftarrow 0] \models \mathcal{I}(\ell'_e)\}$. For player `sched`, we have $\Sigma_{\mathtt{sched}}(\ell,\nu) = \mathfrak{S}(|P|)$. The initial state is $s^c_{\mathsf{init}} = (\iota, \mathbf{0})$. The transitions are defined as follows. Intuitively, each player in P suggests a pair (d,e) of delay and an edge, and player `sched`'s choice determines which player's move is to be taken among those that have suggested the least delay. Formally, we have $\delta((\ell,\nu),(d_1,e_1),\ldots,(d_n,e_n),\pi) = \mathtt{Succ}_{e_{i_0}}((\ell,\nu+d_{i_0}))$ where $i_0 = \mathrm{argmin}_{i\in P: d_i = \min_{j\in P} d_j}\ \pi(i)$.

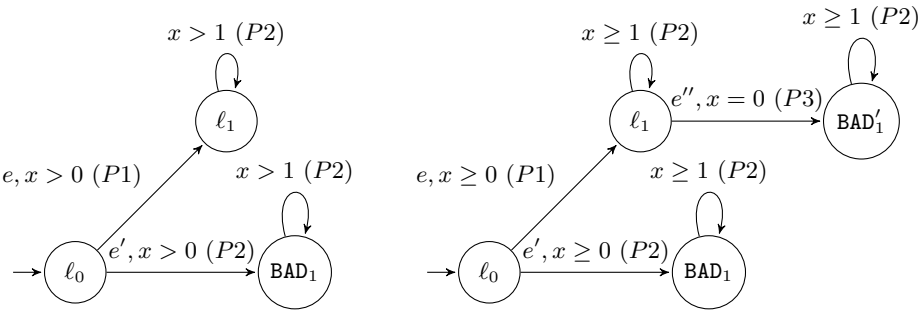

Fig. 1. Two timed games. Invariants are $x \leq 2$ everywhere.

Consider the game on the left in Fig. 1 where the safety objective of player P1 is to avoid location $\mathtt{BAD}_1$. Consider any move (t,e) of P1, the move $(t/2,e)$ dominates (t,e) because any strategy of P2 that plays (t',e') either makes both moves winning if $t' > t$ (or $t = t'$ and P1 is scheduled); either makes both moves losing if $t' < t/2$ (or $t = t'/2$ and P2 is scheduled); either makes $(t/2,e)$ wins and (t,e) loses otherwise. However, it can be seen that $(t/2,e)$ is also dominated by $(t/4,e)$, which is itself dominated, and so on. Thus, there is no admissible strategy in this game.

Here the non-existence of admissible strategy in the continuous time semantics is partly due to the presence of *open guards* (that is, involving strict inequalities only). With these guards,

there is no minimal delay that players can choose. Unfortunately problems also occurs in games with *closed guards*. Consider the game on the right in Fig. 1. The same discussion holds for moves of P1 with positive delays, each such move is dominated by any move with strictly smaller but positive delays. This time there is a unique admissible strategy, the one that plays $(0, e)$. However, the move $(0, e)$ leads to the state $(\ell_1, 0)$ where player P3 can make P1 lose by going to the right, so the unique admissible strategy does not dominate the other strategies. Further, there exist safety conditions for P2 and P3 such that $(0, e)$ is arguably the worst possible move for P1 (e.g. P2 wants to avoid $\mathtt{BAD}_1 \wedge x = 0$ so do not play $(0, e')$ and P3 wants to avoid $\ell_1 \wedge x > 0$ so play $(0, e'')$ in $(\ell_1, 0)$).

4 Admissible Strategies in Discrete Timed Games

Consider a game $\mathcal{D}_M(\mathcal{G})$ for some constant M, and simple safety objectives $(\phi_i)_{i \in P}$. We will only consider simple safety objectives, which are prefix-independent. Therefore, the value of a history only depends on its last state. For each player i, let us partition the state space S of $\mathcal{D}_M$ into $\mathtt{Win}_i = \{s \in S \mid \chi^i_s = 1\}$, $\mathtt{Maybe}_i = \{s \in S \mid \chi^i_s = 0\}$, $\mathtt{Lose}_i = \{s \in S \mid \chi^i_s = -1\}$.

For each player i and history h ending in a state s, a move $m \in \mathsf{M}_i(s)$ is said to be a *winning move* from h if there exists a winning strategy σ for Player i such that h is compatible with σ, and $\sigma(h) = m$.

We introduce the following notations. For any edge $e = (\ell, g, R, \ell')$, and set of states Z, let $\mathtt{Succ}_e(Z) = \{(\ell', \nu') \mid \exists (\ell, \nu) \in Z, \nu \models \mathcal{I}(\ell) \wedge g_e, \nu[R \leftarrow 0] = \nu', \nu' \models \mathcal{I}(\ell')\}$, which is the *immediate successors of Z through edge e*. We define the *immediate predecessors through e* as $\mathtt{Pred}_e(Z) = \{(\ell, \nu) \mid \nu \models g_e \wedge \mathcal{I}(\ell), \exists (\ell', \nu') \in Z, \nu' \models \mathcal{I}(\ell'), \nu[R \leftarrow 0] = \nu'\}$, and *immediate predecessors for Players $I \subseteq P$* as $\mathtt{Pred}_I(Z) = \bigcup_{i \in I, e \in \Delta_i} \mathtt{Pred}_e(Z)$.

Lemma 1 applied to discrete-time semantics of Section 3 gives the following characterisation of dominance of moves in terms of values obtained in case the prescribed move is selected.

Theorem 3. *Consider any player i and state q of $\mathcal{D}_M(\mathcal{G})$. If $q \in \mathtt{Win}_i$, then exactly all winning moves from q are locally admissible. If $q \in \mathtt{Lose}_i$ then all available moves are locally admissible. Assume now that $q \in \mathtt{Maybe}_i$. A move $e \in \Delta_i(q)$ is locally admissible from q if, and only if either $\mathtt{Succ}_e(q) \in \mathtt{Win}_i$ or the following conditions hold*

- $\forall e' \in \mathsf{M}_i(q), \mathtt{Succ}_{e'}(q) \notin \mathtt{Win}_i$,
- $\mathtt{Succ}_e(q) \in \mathtt{Maybe}_i$, *or* $\bot \notin \mathsf{M}_i(q) \wedge \forall e' \in \Delta_i(q), \mathtt{Succ}_{e'} \in \mathtt{Lose}_i$,
- $q +_M 1 \in \mathtt{Win}_i \Rightarrow \exists e' \in \Delta_{-i}(q), \mathtt{Succ}_{e'}(q) \notin \mathtt{Win}_i \wedge \mathtt{Succ}_e(q) \neq \mathtt{Succ}_{e'}(q)$.

Moreover, $\bot$ is locally admissible if, and only if, $\bot \in \mathsf{M}_i(q)$, $\forall e \in \Delta_i(q), \mathtt{Succ}_e(q) \notin \mathtt{Win}_i$ and one of the following conditions holds.

1. $q +_M 1 \in \mathtt{Maybe}_i \cup \mathtt{Win}_i$, *and if* $\exists e \in \Delta_i(q), \mathtt{Succ}_e(q) = q$ *and* $q +_M 1 = q$, *then* $\exists e' \in \Delta_{-i}(q), \mathtt{Succ}_{e'}(q) \notin \mathtt{Lose}_i$, *and* $\mathtt{Succ}_{e'}(q) \neq q$.
2. $\forall e \in \Delta_i(q)$ *such that* $\mathtt{Succ}_e(q) \in \mathtt{Maybe}_i$, *we have that* $\exists e' \in \Delta_{-i}(q)$, *with* $\mathtt{Succ}_{e'}(q) \notin \mathtt{Lose}_i$ *and* $\mathtt{Succ}_{e'}(q) \neq \mathtt{Succ}_e(q)$.

Proof. Let us show that moves satisfying the above properties are locally admissible. We consider history h ending in some state $q \in \mathtt{Maybe}_i$ which is the only non-trivial case.

Let us start with the following simple but useful remark.

Remark 1. Consider any state q, $c \in \mathsf{M}_{-i}(q), e, e' \in \Delta_i(q)$. Then, either $\delta(q, e, c) = \mathtt{Succ}_e(q)$ and $\delta(q, e', c) = \mathtt{Succ}_{e'}(q)$ or $\delta(q, e, c) = \delta(q, e', c)$.

– Consider $e \in \mathsf{M}_i(q)$ where $\mathtt{Succ}_e(q) \in \mathtt{Win}_i$. If $e <_h e'$, by Remark 1 and Lemma 1 item (i), we must have $\mathtt{Succ}_{e'}(q) \in \mathtt{Win}_i$ too. But by the same remark, item (ii) of Lemma 1 cannot hold, which shows $e \not<_h e'$.
– Consider $e \in \mathsf{M}_i(q)$ with $\mathtt{Succ}_e(q) \in \mathtt{Maybe}_i$, and assume $\forall e' \in \mathsf{M}_i(q), \mathtt{Succ}_{e'}(q) \notin \mathtt{Win}_i$.
  - Assume $e <_h e'$. If $\mathtt{Succ}_e(q) = \mathtt{Succ}_{e'}(q)$, then e' cannot dominate e by Remark 1 and Lemma 1 item (ii). Otherwise, by assumption, $\mathtt{Succ}_{e'}(q) \notin \mathtt{Win}_i$. If $\mathtt{Succ}_{e'}(q) \in \mathtt{Lose}_i$, then $e \not<_h e'$ by Lemma 1 item (i). If $\mathtt{Succ}_{e'}(q) \in \mathtt{Maybe}_i$, since $\mathtt{Succ}_e(q) \neq \mathtt{Succ}_{e'}(q)$, we have $e \not<_h e'$ by Lemma 1, item (iii).
  - Assume $\bot \in \mathsf{M}_i(q)$ and $e <_h \bot$.
    – Consider the case $q{+_M}1 \notin \mathtt{Win}_i$. Let $c \in \mathsf{M}_{-i}(q)$ be such that all players wait. Then, $\delta(q, \bot, c) = q{+_M}1 \notin \mathtt{Win}_i$, while $\delta(q, e, c) = \mathtt{Succ}_e(q) \in \mathtt{Maybe}_i$. Assume $q{+_M}1 \neq q$. Then, we also have $q{+_M}1 \neq \mathtt{Succ}_e(q)$, so $e \not<_h \bot$ by Lemma 1, item (iii).
    Assume $q{+_M}1 = q$. If $\mathtt{Succ}_e(q) \neq q$, then we conclude similarly as above since $\mathtt{Succ}_e(q) \in \mathtt{Maybe}_i$. Assume $\mathtt{Succ}_e(q) = q$. Since $q \notin \mathtt{Win}_i$, there must be an edge $e' \in \Delta_{-i}(q)$, such that $q \neq \mathtt{Succ}_{e'}(q) \notin \mathtt{Win}_i$. If $c \in \mathsf{M}_{-i}(q)$ denotes the profile which gives Player i priority, and otherwise chooses e', we have $\delta(q, e, c) = q$, and $\delta(q, \bot, c) = \mathtt{Succ}_{e'}(q) \neq q$. This shows that $e \not<_h \bot$: if $\mathtt{Succ}_{e'}(q) \in \mathtt{Lose}_i$, this follows by item (i) of Lemma 1, and if $\mathtt{Succ}_{e'}(q) \in \mathtt{Maybe}_i$, by item (iii).
    – Consider now the case $q{+_M}1 \in \mathtt{Win}_i$, and let $e' \in \Delta_{-i}(q)$ such that $\mathtt{Succ}_{e'}(q) \notin \mathtt{Win}_i \wedge \mathtt{Succ}_e(q) \neq \mathtt{Succ}_{e'}(q)$. Let $c \in \mathsf{M}_{-i}(q)$ which gives priority to Player i, and otherwise picks e'. We thus have $\delta(q, e, c) \neq \delta(q, \bot, c)$ and neither of them are winning, while $\delta(q, e, c) \in \mathtt{Maybe}_i$. So, by Lemma 1, $e \not<_h \bot$.

Consider the delays. Assume that $\bot \in \mathsf{M}_i(q)$, and $\forall e \in \mathsf{M}_i(q), \mathtt{Succ}_e(q) \notin \mathtt{Win}_i$.

– Assume $q{+_M}1 \in \mathtt{Maybe}_i \cup \mathtt{Win}_i$ and fix $e \in \Delta_i(q)$. If $q{+_M}1 \in \mathtt{Win}_i$ or $\mathtt{Succ}_e(q) \in \mathtt{Lose}_i$, then, we cannot have $\bot <_h e$ by case (i) of Lemma 1. Assume that both belong to $\mathtt{Maybe}_i$. Whenever $q{+_M}1 \neq q$ or $\mathtt{Succ}_e(q) \neq q$, we have $q{+_M}1 \neq \mathtt{Succ}_e(q)$ for any target location and reset set e might have, which entails $\bot \not<_h e$ by Lemma 1, item (iii). Assume now that $q{+_M}1 = q$ and $\mathtt{Succ}_e(q) = q$. In this case, there is $e' \in \Delta_{-i}(q)$, $\mathtt{Succ}_{e'}(q) \in \mathtt{Maybe}_i \cup \mathtt{Win}_i$, $\mathtt{Succ}_{e'}(q) \neq q$. For $c \in \mathsf{M}_{-i}(q)$ which gives priority to Player i, and otherwise chooses e', we have $\delta(q, \bot, c) = \mathtt{Succ}_{e'}(q)$ and $\delta(q, e, c) = q$. If $\delta(q, \bot, c)$ has value 1, then $\bot \not<_h e$ by Lemma 1 item (i); and if it has value 0, $\bot \not<_h e$ follows from Lemma 1, item (iii) since $\mathtt{Succ}_{e'}(q) \neq q$.
– Assume that $\forall e \in \Delta_i(q)$ such that $\mathtt{Succ}_e(q) \in \mathtt{Maybe}_i$, we have that $\exists e' \in \Delta_{-i}(q)$, with $\mathtt{Succ}_{e'}(q) \notin \mathtt{Lose}_i$ and $\mathtt{Succ}_{e'}(q) \neq \mathtt{Succ}_e(q)$.
Assume that $\bot <_h e$. If $\mathtt{Succ}_e(q) \in \mathtt{Lose}_i$, then item (ii) of Lemma 1 cannot be satisfied which contradicts $\bot <_h e$. Suppose that $\mathtt{Succ}_e(q) \in \mathtt{Maybe}_i$, and let $e' \in \Delta_{-i}(q)$ given by the above property. Let $c \in \mathsf{M}_{-i}(q)$ which gives Priority to i, and otherwise chooses e'. We have that $\delta(q, \bot, c) = \mathtt{Succ}_{e'}(q) \in \mathtt{Maybe}_i \cup \mathtt{Win}_i$ while $\delta(q, e, c) = \mathtt{Succ}_e(q) \in \mathtt{Maybe}_i$. If $\mathtt{Succ}_e(q) \in \mathtt{Lose}_i$ or $\mathtt{Succ}_{e'}(q) \in \mathtt{Win}_i$, this contradicts $\bot <_h e$ by item (i) of Lemma 1; and if $\mathtt{Succ}_e(q) \in \mathtt{Maybe}_i$, by item (iii) since $\mathtt{Succ}_e(q) \neq \mathtt{Succ}_{e'}(q)$.

We now show the other direction. We prove that any move that does not satisfy the conditions is locally dominated. Consider any history h ending in q, and $e \in \Delta_i(q)$ that satisfies $\mathtt{Succ}_e(q) \notin \mathtt{Win}_i$ and

$$\begin{array}{l} \exists e' \in \Delta_i(q), \mathtt{Succ}_{e'}(q) \in \mathtt{Win}_i \\ \vee \\ \mathtt{Succ}_e(q) \notin \mathtt{Maybe}_i \wedge (\bot \in \mathsf{M}_i(q) \vee \exists e' \in \Delta_i(q), \mathtt{Succ}_{e'}(q) \in \mathtt{Maybe}_i) \\ \vee \\ q{+_M}1 \in \mathtt{Win}_i \wedge \forall e' \in \Delta_{-i}(q), (\mathtt{Succ}_{e'}(q) \in \mathtt{Win}_i \vee \mathtt{Succ}_e(q) = \mathtt{Succ}_{e'}(q)) \,. \end{array}$$

- Case $\exists e' \in \mathsf{M}_i(q), \mathtt{Succ}_{e'}(q) \in \mathtt{Win}_i$. We have $e <_h e'$ by Remark 1. In fact, if Player i's move is selected given some $c \in \mathsf{M}_{-i}(q)$, from $\delta(q, e, c)$, he can continue with a winning strategy, although from $\delta(q, e', c)$ he can lose. If another player's move is selected, then the successors are identical.
- Case $\mathtt{Succ}_e(q) \notin \mathtt{Maybe}_i \wedge (\bot \in \mathsf{M}_i(q) \vee \exists e' \in \Delta_i(q), \mathtt{Succ}_{e'}(q) \in \mathtt{Maybe}_i)$. This means that $\mathtt{Succ}_e(q) \in \mathtt{Lose}_i$. We distinguish two cases. If such an e' exists, then it is clear that $e <_h e'$ by Remark 1. Let us assume that no such e' exists and $\bot \in \mathsf{M}_i(q)$.
  Define a move $c \in \mathsf{M}_{-i}(q)$ as follows. If there exists $e' \in \Delta_j$ for some $j \in P$ such that $\mathtt{Succ}_{e'}(q) \in \mathtt{Maybe}_i \cup \mathtt{Win}_i$, then $c_j = e'$, $c_k = \bot$ for all $k \neq i, j$; and $\mathtt{sched}$ gives priority to i, and then to j. Notice that all players k can wait since $\bot \in \mathsf{M}_i(q)$. Otherwise, let $\forall k \in P, c_k = \bot$, and $\mathtt{sched}$ is arbitrary.
  Since $q \in \mathtt{Maybe}_i$, we have $\delta(q, \bot, c) \in \mathtt{Maybe}_i \cup \mathtt{Win}_i$ in all cases. We show that $e <_h \bot$ using Lemma 1. Notice that $\chi^i_{\delta(h,e,c)} < \chi^i_{\delta(h,\bot,c)}$ since e moves to $\mathtt{Lose}_i$, and $\chi^i_{\delta(h,\bot,c)} \geq 0$ by the previous remark. This shows (ii). Furthermore, for all $c \in \mathsf{M}_{-i}(q)$, $\chi^i_{h\delta(h,e,c)} \leq \chi^i_{h\delta(h,\bot,c)}$ since either Player i's edge is picked and the inequality is strict, or another move is picked and the successor states are identical in both cases (Remark 1). This shows (i) and (iii), proving $e <_h \bot$.
- Case $q +_M 1 \in \mathtt{Win}_i$ and for all $e' \in \Delta_{-i}(q)$, either $\mathtt{Succ}_{e'}(q) \in \mathtt{Win}_i$ or $\mathtt{Succ}_e(q) = \mathtt{Succ}_{e'}(q)$. Here, we show that $e <_h \bot$. Note that $q +_M 1 \in \mathtt{Win}_i$ means $\bot \in \mathsf{M}_i(q)$. Item (i) of Lemma 1 is satisfied since for all $c \in \mathsf{M}_{-i}(q)$, either $\delta(q, e, c) = \delta(q, \bot, c)$ or $\delta(q, \bot, c) \in \mathtt{Win}_i$. Item (iii) follows from this remark. Moreover, $q +_M 1$ is a possible successor under $\bot$, which shows item (ii).

Consider the move $\bot$ from history h ending in q with $\bot \in \mathsf{M}_i(q)$, such that either $\exists e \in \Delta_i(q), \mathtt{Succ}_e(q) \in \mathtt{Win}_i$ or we have the conjunction of the following:

a) $q +_M 1 \in \mathtt{Lose}_i$, or $q +_M 1 = q$ and $\exists e \in \Delta_i(q)$, $\mathtt{Succ}_e(q) = q$ and $\forall e' \in \Delta_{-i}(q)$, $\mathtt{Succ}_{e'}(q) \in \mathtt{Lose}_i \vee \mathtt{Succ}_{e'}(q) = q$.
b) $\exists e \in \Delta_i(q)$ with $\mathtt{Succ}_e(q) \in \mathtt{Maybe}_i$ and $\forall e' \in \mathsf{M}_{-i}(q), \mathtt{Succ}_{e'}(q) \in \mathtt{Lose}_i \vee \mathtt{Succ}_{e'}(q) = \mathtt{Succ}_e(q)$.

- If $\exists e \in \Delta_i, \mathtt{Succ}_e(q) \in \mathtt{Win}_i$, then $\bot <_h e$. In fact, whenever Player i has priority, the move e yields to a winning state; while if Player i waits, then either a delay or another edge $e' \in \Delta_{-i}(q)$ must yield to a state in $\mathtt{Maybe}_i \cup \mathtt{Lose}_i$ since $q \in \mathtt{Maybe}_i$.
- Consider first the case $q +_M 1 \in \mathtt{Lose}_i$ and b). We show that $\bot <_h e$. For all $c \in \mathsf{M}_{-i}(q)$, by hypothesis, $\delta(q, \bot, c)$ is losing if $\delta(q, \bot, c) = q +_M 1$ or $\delta(q, \bot, c) = \mathtt{Succ}_{e'}(q) \in \mathtt{Lose}_i$ for some $e' \in \Delta_{-i}(q)$. Otherwise, if $\delta(q, \bot, c) = \mathtt{Succ}_{e'}(q) \notin \mathtt{Lose}_i$ then, we must have $\mathtt{Succ}_e(q) = \mathtt{Succ}_{e'}(q)$. This shows items (i) and (iii) of Lemma 1. Moreover, we have item (ii) since when all other players wait, $\delta(q, \bot, c) = q +_M 1 \in \mathtt{Lose}_i$ while $\delta(q, e, c) \in \mathtt{Maybe}_i$.

  Last, assume that $q +_M 1 = q$ and $\exists e \in \Delta_i(q), \mathtt{Succ}_e(q) = q$, and $\forall e' \in \Delta_{-i}(q)$, $\mathtt{Succ}_{e'}(q) \in \mathtt{Lose}_i \vee \mathtt{Succ}_{e'}(q) = q$, and b). This means that $\forall e' \in \Delta_{-i}(q)$, $\mathtt{Succ}_{e'}(q) \in \mathtt{Lose}_i$ or $\mathtt{Succ}_{e'}(q) = q$. Observe also that since $q \in \mathtt{Maybe}_i$, and $q +_M 1 = q$, there must exist $e_0 \in \Delta_{-i}(q)$ with $q \neq \mathtt{Succ}_{e_0}(q) \in \mathtt{Lose}_i \cup \mathtt{Maybe}_i$ since otherwise q would be a winning state. It follows that $\mathtt{Succ}_{e_0}(q) \in \mathtt{Lose}_i$. Let us show $\bot <_h e$. Let $c \in \mathsf{M}_{-i}(q)$. If $\delta(q, \bot, c) \notin \mathtt{Lose}_i$, then $\delta(q, \bot, c) = q$, that is, either $\delta(q, \bot, c) \in \mathtt{Lose}_i$, or $\delta(q, \bot, c) = \delta(q, e, c)$. This shows item (i) and (iii) of Lemma 1. Moreover, if c gives Priority to i, and otherwise chooses e_0, we have $\delta(q, \bot, c) = \mathtt{Succ}_{e_0}(q) \in \mathtt{Lose}_i$ and $\delta(q, e, c) = q \in \mathtt{Maybe}_i$, which shows item (ii).

□

5 Computation using Zones

We assume that clocks are bounded in all locations by an invariant:

Assumption 4 *In all considered timed games, the invariant at each location implies* $\bigwedge_{x\in X} x < M$.

5.1 Zones and Difference-Bound Matrices

Formally, a *zone* Z is a convex subset of $\mathbb{R}^X_{\geq 0}$ definable by a conjunction of constraints of the form $x \bowtie k$, $l \bowtie x$, or $x - y \bowtie m$ where $x, y \in X$, $k, l \in \mathbb{N}_{\geq 0}$, $m \in \mathbb{Z}$, and $\bowtie \in \{<, \leq\}$.

We recall a few basic operations defined on zones. Let $Z\uparrow$ denote the time-successors of Z, i.e., $Z\uparrow = \{\nu \in \mathbb{R}^X_{\geq 0} \mid \exists \nu' \in Z, \exists t \geq 0, \nu = \nu' + t\}$; and similarly the time-predecessors are $Z\downarrow = \{\nu \in \mathbb{R}^X_{\geq 0} \mid \exists t \geq 0, \nu + t \in Z\}$. For $R \subseteq X$, we define $Z[R \leftarrow 0] = \{\nu \in \mathbb{R}^X_{\geq 0} \mid \exists \nu' \in Z, \nu = \nu'[R \leftarrow 0]\}$, and $\mathtt{Free}_R(Z) = \{\nu \in \mathbb{R}^X_{\geq 0} \mid \nu[R \leftarrow 0] \in Z\}$. Intersection is denoted $Z \cap Z'$. It is well known that zones are closed under all these operations [6].

Zones can be represented by *difference-bound matrices (DBM)* which are $|X_0| \times |X_0|$-matrices with values in $\mathbb{Z} \times \{<, \leq\}$ [22], where $X_0 = X \cup \{0\}$. Here 0 is seen as a clock whose value is always 0. Intuitively, each component $(x, y) \in X_0 \times X_0$ of a DBM stores a bound on the difference $x - y$. We use the following notations to access to components of a DBM D. For $x, y \in X_0$, let the component (x, y) be written as $(D_{x,y}, \prec^D_{x,y})$. For any DBM D, let $[\![D]\!]$ denote the zone it defines. The DBM D is *reduced* if no constraint can be made tighter without changing the defined zone. This is true when the following inequalities are satisfied: for all $x, y, z \in X_0$, $(D_{x,y}, \prec^D_{x,y}) < (D_{x,z}, \prec^D_{x,z}) + (D_{z,y}, \prec^D_{z,y})$ where we define $(a, \prec) + (b, \prec') = (a+b, \prec'')$ with $\prec''=<$ if, and only if $\prec=<$ or $\prec'=<$; while $(a, \prec) < (b, \prec')$ if $a < b$ or $a = b$ and either $\prec=\prec'$ or $\prec=<$ and $\prec'=\leq$. Every DBM can be made reduced using shortest path algorithms. We refer the reader to [6] for details on operations on DBMs.

We define an *extended DBM* as a pair (ℓ, Z) where ℓ is a location and Z a zone. Let $[\![(\ell, Z)]\!]$ denote the set $\{\ell\} \times [\![Z]\!]$. By a slight abuse of notation, we will use the same operations for DBMs as for zones, for instance, we will write $D' = D\uparrow$ where D and D' are DBMs such that $[\![D']\!] = [\![D]\!]\uparrow$. In this case, D' can be computed using algorithms described in [6]. Successors and predecessors of zones are zones as well, and can be computed efficiently using DBMs. Let us consider an extended state (ℓ, Z) and an edge $e = (\ell, g, R, \ell')$. We define $\mathtt{Succ}_e((\ell, Z)) = \cup_{q \in [\![(\ell,Z)]\!]} \mathtt{Succ}_e(q)$, and $\mathtt{Pred}_e((\ell', Z)) = \{\ell\} \times \{\nu \mid \nu \models g, \nu[R \leftarrow 0] \in [\![Z]\!]\}$. A DBM (resp. a zone) is closed if the set it defines is topologically closed. Equivalently, a closed DBM can be defined using a conjunction of non-strict constraints.

A *federation* is a list of DBMs $F = \cup_i D_i$, and defines the set $[\![F]\!] = \cup_i [\![D_i]\!]$. We define the complement of a zone Z in $\mathbb{R}^X_{\geq 0}$ as $\overline{Z}$. If a zone is represented as a DBM D, its complement can be computed as a federation, denoted $\overline{D}$; that is, $[\![\overline{D}]\!] = \overline{[\![D]\!]}$. By extension, we also call an *extended federation* a union of extended zones. Given an extended federation F and location ℓ, we denote by $\ell \cap F$ the set $\{\ell\} \times \mathbb{R}^X_{\geq 0} \cap F$; thus, each location ℓ denotes an extended zone at that location with no constraint on clocks. A *closed federation* is a federation whose DBMs are closed. We extend all operations on DBMs to federations by applying them on all elements of the federation. For instance, $F\downarrow = \cup_i D_i\downarrow$; while intersection is defined by $(\cup_i D_i) \cap (\cup_j D'_j) = \cup_{i,j} D_i \cap D'_j$. For the complement, we set $\overline{F} = \cap_i \overline{D_i}$.

In order to consider the discrete-time semantics, let us define $[\![Z]\!]_{\mathtt{d}} = \{\nu \in \mathbb{N}^{X_0} \mid \nu \models Z\}$. In other terms, $[\![Z]\!]_{\mathtt{d}} = \mathbb{N}^{X_0} \cap [\![Z]\!]$. Given any DBM D, let $\mathtt{closed}(D)$ denote the largest closed zone contained in D. Formally, we have $D' = \mathtt{closed}(D)$ where for all i, j, $D_{i,j} = D'_{i,j}$ if the latter is a non-strict constraint, and $D_{i,j} = (a - 1, \leq)$ if $D'_{i,j} = (a, <)$. Intuitively, the $\mathtt{closed}(D)$ returns

a closed DBM whose discrete valuations are identical to those of D. Notice that $\texttt{closed}(D)$ can be empy although D is not. For any zone Z, let $\overline{Z}^{\mathsf{d}} = \mathbb{N}^X \setminus Z$; and we extend this notation to DBMs and federations. We also define discrete time-successors as $Z{\uparrow}^{\mathsf{d}} = \{\nu + d \mid d \in \mathbb{N}, \nu \in Z\}$, and $Z{\downarrow}^{\mathsf{d}} = \{\nu \mid \exists d \in \mathbb{N}, \nu + d \in Z\}$. Similarly, let $\texttt{Free}^{\mathsf{d}}_R(Z) = \{\nu \in \mathbb{N}^X \mid \nu[R \leftarrow 0] \in Z\}$.

Lemma 2. *Let Z, Z' be DBMs and $R \subseteq X$. The following properties hold.*

- $[\![\texttt{closed}(Z)]\!]_{\mathsf{d}} = [\![Z]\!]_{\mathsf{d}}$,
- $\overline{[\![Z]\!]_{\mathsf{d}}}^{\mathsf{d}} = [\![\texttt{closed}(\overline{Z})]\!]_{\mathsf{d}}$,
- $[\![Z]\!]_{\mathsf{d}} \cap [\![Z']\!]_{\mathsf{d}} = [\![Z \cap Z']\!]_{\mathsf{d}}$,
- $\texttt{Free}^{\mathsf{d}}_R([\![Z]\!]_{\mathsf{d}}) = [\![\texttt{Free}_R(Z)]\!]_{\mathsf{d}}$,
- *if Z is closed,* $[\![Z]\!]_{\mathsf{d}}[R \leftarrow 0] = [\![Z[R \leftarrow 0]]\!]_{\mathsf{d}}$,
- *if Z is closed,* $[\![Z]\!]_{\mathsf{d}}{\downarrow}^{\mathsf{d}} = [\![Z{\downarrow}]\!]_{\mathsf{d}}$,
- *if Z is closed,* $[\![Z]\!]_{\mathsf{d}}{\uparrow}^{\mathsf{d}} = [\![Z{\uparrow}]\!]_{\mathsf{d}}$.

Closed federations are closed under all above operations.

Thanks to the above lemma, we will represent sets of discrete states using DBMs. Intuitively, we let a closed zone represent the set of discrete valuations it contains, while the lemma ensures that basic operations applied on the zone corresponds to the corresponding operations in the discrete semantics.

Note that all operations but complementation are continuous, thus preserve closedness. Since all guards are closed in the discrete-time setting, the successor and predecessor operators are defined identically to the continuous case, without using the `closed` operator. That is, given extended zone (ℓ, Z) where Z is represented by closed DBM D, and edge $e = (\ell, g, R, \ell')$, $[\![\texttt{Succ}_e((\ell, Z))]\!]_{\mathsf{d}} = \{(\ell', \nu') \in \mathcal{L} \times \mathbb{N}^X \mid \nu \models g \wedge \mathcal{I}(\ell), \nu[R_e \leftarrow 0] = \nu' \models \mathcal{I}(\ell')\}$, while this set can be computed by $(((D \cap g_e \cap \texttt{Inv}(\ell))[R \leftarrow 0]) \cap \texttt{Inv}(\ell'))\uparrow \cap \texttt{Inv}(\ell')$, where $\texttt{Inv}$ is the extended federation defining for each ℓ the invariant $\texttt{Inv}(\ell)$ at location ℓ. The predecessors are computed similarly by $(\texttt{Free}_R(Z \cap \texttt{Inv}(\ell') \cap (R = 0)) \cap g_e \cap \texttt{Inv}(\ell))\downarrow \cap \texttt{Inv}(\ell)$.

5.2 Computing State Values

We show how to use a zone-based exploration to compute state values for each player. As in the previous section, we consider the extended federation $\texttt{Inv}$ which defines the set $\mathcal{I}$ of states that satisfy their locations' invariants, that is, $\mathcal{I} = [\![\texttt{Inv}]\!] = \cup_{\ell \in \mathcal{L}} \ell \times [\![\texttt{Inv}(\ell)]\!]$.

Given $B, G \subseteq S$. Let $\texttt{TPred}^{\mathsf{d}}_i(G, B) = \{q \in S \mid \exists d \in \mathbb{N}, q + d \in G, q + [0, d] \cap B = \emptyset\}$. This is the set of states which, by a discrete delay, can reach G while avoiding B in during the delay. One could also define $\texttt{TPred}^{\mathsf{d}}_i$ by fixing a unit delay $d = 1$, and repeating it. However, quantifying over $d \in \mathbb{N}$ will allow us to use DBMs to compute this operator efficiently.

We define $\pi^{\mathsf{d}}_i(Z) = \texttt{TPred}^{\mathsf{d}}_i(\texttt{Pred}_i(Z), \texttt{Pred}_{-i}(\overline{Z}^{\mathsf{d}}))$. Let us thus first state the set theoretic fixpoint defining the winning region in the discrete semantics. Below, ν is the greatest fixpoint operator; we will also use the least fixpoint operator μ.

Lemma 3. *For any timed game $\mathcal{G}$, player i, bad states B_i, we have* $\texttt{Win}_i = \nu Z. \overline{B_i}^{\mathsf{d}} \cap S \cap \pi^{\mathsf{d}}_i(Z)$.

When G and B are federations, we write $\texttt{TPred}^{\mathsf{d}}_i(G, B) = \texttt{TPred}^{\mathsf{d}}_i([\![G]\!]_{\mathsf{d}}, [\![B]\!]_{\mathsf{d}})$, and $\pi^{\mathsf{d}}_i(G) = \pi^{\mathsf{d}}_i([\![G]\!]_{\mathsf{d}})$. The following lemma is adapted from [17, Lemma 4]

Lemma 4. *Consider any timed game $\mathcal{G}$, player i, bad states B. Given closed federations $G = \cup_k G_k$ and $B = \cup_j B_j$ both contained in* $\texttt{Inv}$, $\texttt{TPred}(G, B)$ *can be computed as follows.* $\texttt{TPred}^{\mathsf{d}}(G, B) = \bigcup_k \bigcap_j \texttt{TPred}^{\mathsf{d}}(G_k, B_j)$, *where* $\texttt{TPred}^{\mathsf{d}}(G_k, B_j) = [\![\texttt{Inv} \cap \Big((G_k{\downarrow} \cap \overline{B_j{\downarrow}}^{\mathsf{d}}) \cup \Big(G_k \cap (B_j{\downarrow}) \cap \overline{B_j}^{\mathsf{d}}\Big){\downarrow}\Big)]\!]_{\mathsf{d}}$.

It follows from Lemma 2 and 4 that given a closed federation F, $\pi_i^{\mathsf{d}}(F)$ can be computed as a closed federation. Thus, $\mathtt{Win}_i$ can be computed as an extended closed federation. The next lemma will show that $\mathtt{Maybe}_i$ can also be computed as an extended closed federation.

We define the discrete variant of $\mathtt{Pred}$ by $\mathtt{Pred}_I^{\mathsf{d}}(Z) = \cup_{i\in I, e\in\Delta_i}\mathtt{Pred}_e^{\mathsf{d}}(Z)$, where $\mathtt{Pred}_e^{\mathsf{d}}(Z) = \{q \in S \mid \mathtt{Succ}_e(q) \in Z\}$.

Lemma 5. *For any timed game $\mathcal{G}$, player i, bad states B_i, we have* $\mathtt{Maybe}_i = \nu Z.\overline{B_i}^{\mathsf{d}} \cap S \cap \mathtt{Pred}_P^{\mathsf{d}}(Z)\downarrow^{\mathsf{d}}$.

Last, the set $\mathtt{Lose}_i$ can be computed as the complement of $\mathtt{Win}_i \cup \mathtt{Maybe}_i$.

We thus showed, in this section, that sets of states with a given value can be computed using federations.

6 Model Checking Under Admissibility

In this section, we show how to check whether all states reachable under admissible strategy profiles satisfy a given invariance property. Formally, the problem is stated as follows.

Problem 1 (Model Checking Under Admissibility). Given a timed game $\mathcal{G}$, simple safety objectives $(\phi_i)_{i\in P}$, and (arbitrary) safety property ϕ, check

$$\forall \boldsymbol{\sigma} \in \prod_{i\in P} \mathtt{Adm}_i(\mathcal{D}_M(\mathcal{G})), \forall \sigma_{\mathtt{sched}} \in \Gamma_{\mathtt{sched}}, \mathtt{Out}(\mathcal{D}_M, \boldsymbol{\sigma}, \sigma_{\mathtt{sched}}) \models \phi.$$

We describe a forward exploration algorithm using federations similar to the usual reachability algorithm except that both discrete transitions and time delays are modified so that only locally admissible moves are considered by players.

For $I \subseteq P$, let $\mathtt{Trap}_I(Z) = \cap_{i\in I}\overline{\mathtt{Pred}_i(\overline{Z}^{\mathsf{d}})}^{\mathsf{d}}$, that is, the set of states from which no player in I can avoid the set Z by choosing a move.

For a reduced DBM D, and $a, b \in \mathbb{N}$, define $\mathtt{Shift}_{a,b}(D) = D'$ as $D'_{i,j} = D_{i,j}$ for all $i, j \neq 0$; $D'_{i,0} = D_{i,0} + b$ and $D'_{0,i} = D_{0,i} - a$ for all $i \neq 0$. Notice that the resulting DBM D' may no more be reduced, so it must be made reduced.

Lemma 6. *Let D be a reduced DBM.*

- $[\![\mathtt{Shift}_{-1,-1}(D)]\!]_{\mathsf{d}} = \{\nu \in \mathbb{N}^X \mid \nu + 1 \in [\![D]\!]_{\mathsf{d}}\}$,
- $[\![\mathtt{Shift}_{0,-1}(D)]\!]_{\mathsf{d}} = \{\nu \in [\![D]\!]_{\mathsf{d}} \mid \nu + 1 \in [\![D]\!]_{\mathsf{d}}\}$,
- $[\![\mathtt{Shift}_{-1,0}(D)]\!]_{\mathsf{d}} = \{\nu \in \mathbb{N}^X \mid \nu \in [\![D]\!]_{\mathsf{d}} \vee \nu + 1 \in [\![D]\!]_{\mathsf{d}}\}$,
- $[\![\mathtt{Shift}_{0,1}(D)]\!]_{\mathsf{d}} = \{\nu \mid \nu \in [\![D]\!]_{\mathsf{d}} \vee \nu - 1 \in [\![D]\!]_{\mathsf{d}}\}$.

Constrained Guards For any location ℓ and edge $e \in \Delta_i(\ell)$, let us define

$$W_e = \{(\ell, \nu) \in \mathtt{Win}_i \mid \nu \models g_e, \mathtt{Succ}_e((\ell, \nu)) \in \mathtt{Win}_i, \mathtt{Succ}_{-i}((\ell, \nu)) \in \mathtt{Win}_i\}.$$

In other terms, W_e is the set of states from which Player i can pick the transition $e \in \Delta_i$ which guarantees staying in $\mathtt{Win}_i$.

Given two edges $e_i = (\ell_i, g_i, R_i, \ell'_i)$ for $i = 1, 2$, let us define the expression $\mathtt{Eq}(e_1, e_2) \equiv (\wedge_{x\in X}((x \in R_1 \cap R_2) \vee x = 0)) \wedge \ell'_1 = \ell'_2$. In other terms, $\mathtt{Eq}(e_1, e_2)$ are the set of states from which the successors through these edges are identical. Let us define

$$g'_e = g_e \wedge (\mathtt{Win}_i \Rightarrow W_e) \wedge (\mathtt{Maybe}_i \Rightarrow \mathtt{Pred}_e(\mathtt{Win}_i) \vee g''_e)\,, \tag{1}$$

where

$$g''_e = \mathtt{Trap}_i(\mathtt{Lose}_i \cup \mathtt{Maybe}_i) \wedge \Big(\mathtt{Pred}_e(\mathtt{Maybe}_i) \cup \overline{\mathtt{Shift}_{0,-1}(\mathtt{Inv})}^{\mathsf{d}} \wedge \mathtt{Trap}_i(\mathtt{Lose}_i)\Big) \wedge \big(\mathtt{Shift}_{-1,-1}(\mathtt{Win}_i) \Rightarrow (\vee_{e' \in \Delta_{-i}} \neg \mathtt{Eq}(e,e') \wedge \overline{\mathtt{Pred}_{e'}(\mathtt{Win}_i)})\big).$$

Lemma 7. *For any player $i \in P$, an edge $e = (\ell, g, R, \ell') \in \Delta_i$ is locally admissible at (ℓ, ν) if, and only if $\nu \models g'_e$.*

The proof follows immediately from Theorem 3.

Constrained Time Successors For each location ℓ, and edge $e \in \Delta_i(\ell)$, define $G_e = \ell \wedge g_e$. For each player i, we define A^i, as the set of states from which waiting is locally admissible, as follows.

$$A^i = \mathtt{Shift}_{0,-1}(\mathtt{Inv}) \cap \Big(\mathtt{Lose}_i \cup \mathtt{Shift}_{0,-1}(\mathtt{Win}_i) \cup \Big(\mathtt{Maybe}_i \cap \overline{\mathtt{Pred}_i(\mathtt{Win}_i)} \cap (B^i \cup C^i)\Big)\Big), \tag{2}$$

where $B^i = \mathtt{Shift}_{-1,-1}(\mathtt{Win}_i \cup \mathtt{Maybe}_i)$ and

$$C^i = \bigcap_{e \in \Delta_i} \left(G_e \cap \mathtt{Pred}_e(\mathtt{Maybe}_i) \Rightarrow \bigcup_{e' \in \Delta_{-i}} G_{e'} \cap \overline{\mathtt{Pred}_{e'}(\mathtt{Lose}_i)} \cap \neg \mathtt{Eq}(e,e') \right).$$

Lemma 8. *Consider state q, and player $i \in P$. Move $\perp$ is locally admissible at q for player i if, and only if $q \in [\![A^i]\!]_{\mathsf{d}}$.*

Let $A = \cap_{i \in P} A^i$. Given set F, let $F{\uparrow}^{\mathsf{d}}_A = \{q \in S \mid \exists q' \in F, d \in \mathbb{N}, q' + d = q, q' + [0, d-1] \subseteq A\}$. Hence, this is the set of time successors of F which are reachable by staying inside A (except that the last state can be outside of A). Notice that all states of $F{\uparrow}^{\mathsf{d}}_A$ satisfy the invariants.

Lemma 9. *For any set F, $F{\uparrow}^{\mathsf{d}}_A = \mu Z.F \cup \mathcal{I} \cap \mathtt{Shift}_{0,1}(Z \cap A)$.*

Algorithm 1 Model checking under admissibility algorithm for safety properties

Input: Game $\mathcal{G}$, simple safety objectives $(\phi_i)_{i \in P}$, $M \in \mathbb{N}$, safety property ϕ
Let $\mathtt{Win}_i = \nu Z.\phi_i \cap \pi^{\mathsf{d}}_i(Z)$, $\mathtt{Maybe}_i = \nu Z.\phi_i \cap \pi^{\mathsf{d}}_P(Z)$, $\mathtt{Lose}_i = \overline{\mathtt{Win}_i \cup \mathtt{Maybe}_i}$
For all $i \in P$, let $\Delta'_i = \{(\ell, g', R, \ell') \mid (\ell, g, R, \ell') \in \Delta_i\}$ where g' is defined (1).
Define A as in (2)
$\mathtt{Waiting} = \{(\ell_0, \mathbf{0})\}$
$\mathtt{Passed} = \emptyset$
while $\mathtt{Waiting} \neq \emptyset$ **do**
  Let $Z = \mathsf{Pop}(\mathtt{Waiting})$
  if $Z \not\models \phi$ **then**
    return False
  $\mathtt{Passed} = \mathtt{Passed} \cup \{Z\}$
  for all $i \in P, e \in \Delta'_i$ **do**
    $Z' = \mathtt{Succ}_{e'}(Z){\uparrow}^{\mathsf{d}}_A$
    if $\neg \exists Z'' \in \mathtt{Passed} \cup \mathtt{Waiting}, Z' \subseteq Z''$ **then**
      $\mathtt{Waiting} = \mathtt{Waiting} \cup \{Z'\}$
return True

Exploration Now during the exploration, given any federation F in the waiting list, and edge e, we expand the search by $\mathtt{Succ}_{e'}(F)$ where e' is the edge e whose guard g_e is replaced by g'_e. We then compute its constrained time successors by restricting the delays to those states A where all players can indeed wait. The algorithm is summarized in Algorithm 1. Notice that Assumption 4 allows us to ensure the termination of the algorithm without using extrapolation operators.

7 Assume-Admissible Synthesis

We now show how to solve the assume-admissible synthesis problem.

Problem 2 (Assume-Admissible Synthesis). Given a timed game $\mathcal{G}$, simple safety objectives $(\phi_i)_{i\in P}$, check if for each player i,

$$\exists \sigma_i \in \mathtt{Adm}_i(\mathcal{D}_M(\mathcal{G})), \forall \sigma_{-i} \in \mathtt{Adm}_{-i}(\mathcal{D}_M(\mathcal{G})), \forall \sigma_{\mathtt{sched}} \in \Gamma_{\mathtt{sched}}, \mathtt{Out}(\mathcal{D}_M, \boldsymbol{\sigma}, \sigma_{\mathtt{sched}}) \models \phi_i.$$

If for each player i, we manage to find an admissible strategy that is winning against all admissible strategy profiles of $-i$, then the combination of these strategies is a profile that satisfies all objectives. Note that players can choose their strategies arbitrarily among these winning ones without coordination with other players as long as other players choose admissible strategies. Let us call σ_i *assume-admissible-winning (AA-winning)* if it witnesses the above condition for Player i.

We are going to solve this problem by using the results of the previous section. In fact, we showed how to strengthen the guards of the edges of the timed automaton so that they are only taken by respective players if the corresponding move is locally admissible. We also characterized those states from which a delay is locally admissible for all players. It remains to show how to solve the game where all players are restricted to behave admissibly.

Given a timed game $\mathcal{G}$, let $\mathcal{G}'$ be obtained by strengthening the guards of all edges as in the previous section. Let A be the set from which waiting is locally admissible for all players, as defined in the previous section. For any set F of states, let us define the *A-constrained time-predecessors* as $F{\downarrow}^{\mathsf{d}}_A = \{q \in S \mid \exists d \in \mathbb{N}, q+d \in F, q+[0,d-1] \subseteq A\}$. Intuitively, this is precisely the set of states which can reach F by time delays while staying in A (except at the last state). This operator can be computed as follows.

Lemma 10. *For all sets $F \subseteq \mathbb{R}^X_{\geq 0}$, $F{\downarrow}^{\mathsf{d}}_A = \mu Z.F \cup (A \cap \mathtt{Shift}_{-1,0}(Z \cap \mathcal{I}))$.*

We define $\mathtt{TPred}^{\mathsf{d}}_{A,i}(G,B) = \{q \in S \mid \exists d \in \mathbb{N}, q+d \in G, q+[0,d] \cap B = \emptyset, q+[0,d-1] \subseteq A\}$. This defines the set of states from it is admissible for players to wait until reaching G while avoiding B.

Lemma 11. *Consider any timed game $\mathcal{G}$, player i, bad states B. Given closed federations $G = \cup_k G_k$ and $B = \cup_j B_j$ both implying $\mathtt{Inv}$, $\mathtt{TPred}^{\mathsf{d}}_{A,i}(G,B)$ can be computed as follows.* $\mathtt{TPred}^{\mathsf{d}}_{A,i}(G,B) = \bigcup_k \bigcap_j \mathtt{TPred}^{\mathsf{d}}_{A,i}(G_k,B_j)$, *where*

$$\mathtt{TPred}^{\mathsf{d}}_{A,i}(G_k,B_j) = [\![\mathtt{Inv} \cap \left((G_k{\downarrow}^{\mathsf{d}}_A \cap \overline{B_j{\downarrow}^{\mathsf{d}}_A}^{\mathsf{d}}) \cup \left(G_k \cap (B_j{\downarrow}^{\mathsf{d}}_A) \cap \overline{B_j}^{\mathsf{d}} \right) {\downarrow}_A \right)]\!]_{\mathsf{d}}.$$

Let $\pi^{\mathsf{d}}_{i,A}(Z) = \mathtt{TPred}^{\mathsf{d}}_{A,i}(\mathtt{Pred}_i(Z), \mathtt{Pred}_{-i}(\bar{Z}))$.

Theorem 5. *Let $\mathcal{G}$ be a game with simple safety objectives $(\phi_i)_{i\in P}$. Let $\mathcal{G}'$ be obtained by replacing each guard g_e by g'_e as defined in (1). Let $W_i = \nu Z.\phi_i \cap S \cap \pi^{\mathsf{d}}_{i,A}(Z)$ computed in $\mathcal{G}'$. Then, Player i has a AA-winning strategy in $\mathcal{G}$ if, and only if the initial state belongs to W_i.*

8 Example: Synthesis of Train Controllers

We consider a one-way circular train network with n segments, and K trains. Each segment models either a station, or a part of the line between two stations. For safety reasons, each segment can accomodate at most one train. In order to optimize performance criteria, trains are allowed to independently regulate their travel time at each segment as long as they meet this safety critical requirement.

Model We describe the system as a network of timed automata defining a discrete timed game with Boolean variables. Each segment j is modeled as in Figure 2: it can be "occupied" by a train upon receiving event m_j, after which it is "freed" by the occupation of the next segment, by event m_{j+1}. The clock y_j stores the time elapsed since the latest train leaving.

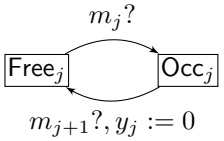

Fig. 2. Component for segment j

Each train i is modeled as a separate player, and its moves are defined by the component in Fig. 3. If the current state of the component is $s_{i,j}$ or $s'_{i,j}$, this means that the train i is at segment j. The train can attempt to move to segment $j+1$ by sending event m_{j+1} if it has spent at least 10 seconds in the current segment. This lower bound corresponds to the minimum travel time (with maximal speed) of a train over a segment. If 30 seconds have elapsed in a given segment, the train either has to move to the next segment, or enters state $s'_{i,j}$ from which at least one unit of time will elapse and the variable $\texttt{err}_i$ will be set. In our model, the segments are passive, and they only react to actions received by trains.

For better readability, we use a particular synchronization semantics: we assume that an event m_i is only possible if three components synchronize on the action. That is, if a train enters from segment j to segment $j+1$, then the train sends $m_{j+1}!$, upon which the first segment is freed by $m_{j+1}?$, and the second one is occupied by $m_{j+1}?$.

Each train controls the edges of its automaton, and the edges of the segments are only taken in synchronization with trains' edges. Thus, each transition in the overall system is controlled by a unique player (i.e. train).

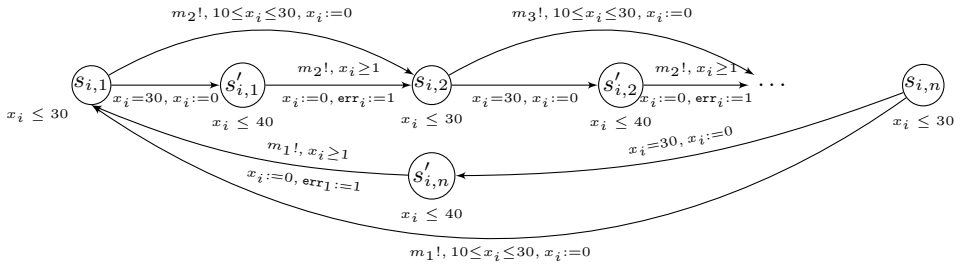

Fig. 3. Component for train i whose edges are controlled by Player i. In addition to the transitions shown in the figure, we add the self-loops with no resets at each state $s_{i,j}$ and $s'_{i,j}$, with the following guard: $\bigvee_{j,k}(s_{j,k} \wedge x_j = 30 \vee s'_{j,k} \wedge x_j = 40)$. In fact, if some train j reaches the upper bound of its invariant, then train i can still choose this self-loop to remain in the current segment.

We define the initial state by assigning each train i to an arbitrary segment j while respecting mutual exclusion: no pair of trains can be at the same segment. Moreover, the segment j is at state Occ_j if, and only if some train is in state s_j. All clocks are initially 0.

Specification Our overall objective Φ is that each segment is served at least once every 150 seconds; in other words, the clocks y_j never exceed 150. Let us thus write $\Phi = \wedge_j \mathsf{G}(y_j \leq 150)$. It is clear that this is not the case in general: if a train stops moving, the following segment is never served. However, we also know that trains do not behave arbitrarily. In fact, to guarantee acceptable passanger experience, each train is also required not to stay for more than 30 seconds at each segment. Let us define $\phi_i = \mathsf{G}(\neg \mathtt{err}_i \wedge x_i \leq 30)$, which is the local specification of train i, that is, its objective. Notice that ϕ_i is a simple safety objective: once $\mathtt{err}_i$ is set to 1, it remains so. Moreover, if x_i exceeds 30, the train is necessarily at some state $s'_{i,j}$ at which the variable $\mathtt{err}_i$ will be set to 1 simultaneously when x_i is reset.

We assume that each train regulates its travel time with the restriction of behaving admissibly with respect to its objective ϕ_i. Now, assuming each train i is admissible for objective ϕ_i, does the global objective Φ hold under all induced executions? This is precisely a model checking under admissibility problem.

Admissible Strategies We have $\mathtt{Win}_i = \emptyset$, $\mathtt{Lose}_i = \mathtt{err}_i \vee (\vee_j s'_{i,j})$, and $\mathtt{Maybe}_i$ is the rest of the states, that is, $\neg \mathtt{err}_i \wedge (\vee_j s_{i,j})$. We have already explained why there are no winning states in the system. To see that all states satisfying $\neg \mathtt{err}_i \wedge x_i \leq 30$ have value 0, consider such a state where train i is at segment j. If station $j+1$ is free, then train i can move as soon as $x_i \geq 10$. Otherwise, if $i+1$ is the index of the train at segment $j+1$, then $x_i \leq x_{i+1}$ since train $i+1$ must have entered segment $j+1$ before train i has entered segment j. In this case, all segments and $n-1$ trains in the network, all trains that are blocking the way to train i can wait until their clocks reach 10, and simultaneously move one after the other to free segment $j+1$. At this point, train i can (wait and) move to segment $j+1$. By repeating this argument, one can construct a run in which train i satisfies its specification.

Let us now apply Theorem 3 to describe locally admissible moves from states of $\mathtt{Maybe}_i$. Notice that the successor of any state of $\mathtt{Maybe}_i$ through edges of type $s_{i,j} \to s_{i,j+1}$ leads to $\mathtt{Maybe}_i$. Since there is no winning states, all these edges are locally admissible according to the theorem. On the other hand, an edge $s_{i,j} \to s'_{i,j}$ is only locally admissible if $x_i = 30$ and no other edge is available. Moreover, any delay at states $s_{i,j}$ is locally admissible as long as the delay is allowed (that is, $\forall i, x_i \leq 29$). Last, from $\mathtt{Lose}_i$, any move is locally admissible.

Model Checking Under Admissibility At any moment, the trains form several blocks of consecutive occupied segments. By the previous description of the locally admissible moves, it follows that the train at the head of each block must eventually move to the next segment before its clock exceeds 30, thus allowing the previous train to move as well. One shows by induction on n that all trains move before their clocks exceed 30, thus along all runs with locally admissible moves, all objectives ϕ_i are satisfied.

Now, the satisfaction of Φ depends on the parameters n and K. One can see that Φ is satisfied as long as $K \geq n-4$. In fact, if there are four consecutive segments at any time, each segment will be entered and left by a train within 150 time units. The specification fails however, when $K < n-4$. This can also be determined by Algorithm 1 applied to the game described above with specifications $(\phi_i)_i$ for the trains, and the global safety property Φ.

Assume-Admissible Synthesis Rather than checking whether *all* executions under admissible strategies satisfy the specification, let us now apply assume-admissible synthesize to synthesize an admissible strategy for each train satisfying its objective against all admissible strategies. One solution to the AA-synthesis is to let trains move to the next segment *as soon as possible*, that is, whenever the following segment is free, and the guards allow them to move. Let σ_i^{ASAP} denote

this strategy. According to the previous paragraph, σ_i^{ASAP} is admissible since it only chooses locally admissible moves. Moreover, it ensures ϕ_i against all admissible strategies of the other trains since we saw that all executions under admissible strategy profiles satisfy ϕ_i.

Thus, for each i, the particular strategy σ_i^{ASAP} is admissible and ensures ϕ_i. What are possible outcomes under the profile $(\sigma_1^{\text{ASAP}}, \ldots, \sigma_n^{\text{ASAP}})$? Observe that all trains move to the next segment whenever their clocks reach 10 (in fact, this is true for the train at the head of its block, and this shows that other trains will also move at the same time). Thus, each train moves to the next segment every 10 time units under this profile. This means that the specification Φ actually holds when $K \geq n - 14$. In fact, given any block of at most 14 unoccupied consecutive segments, each of them will be served by a train in at most $(14 + 1) \times 10$ time units. Hence, we have synthesized a particular admissible strategy profile in which, not only, each train ensures its own specification against *all* admissible strategy profiles of other trains, but moreover, together, the strategy profile satisfies the specification Φ for a larger choice of parameters K and n.

Discussion We showed that all admissible strategies satisfy the minimal performance requirement Φ in our system (given constraints on the parameters K, n). Thus, an admissible strategy for each train can be chosen separately according to other given performance criteria if desired, and Φ will hold regardless of the precise choice.

We thus suggest a two-step synthesis methodology where we separate minimal performance requirement Φ from further optimization criteria. We ensure this first step formally using admissibility, while the further steps can be done using other methods: above, we used assume-admissible synthesis, but other methods can be used as well such as statistical learning with the only requirement of being compatible with locally admissible moves. Thus, one is able to formally ensure strong guarantees for Φ, and use other methods with relaxed guarantees for further optimization.

We kept the model extremely simple in order to make it human-readable. Several details can be added to approach a more realistic model. First, the topology of the network can be made arbitrary, and two-way traffic can be incorporated with possible shared (thus, mutually exclusive) portions between different lines. Most importantly, perturbations in travel times can be added by introducing a player which adds a bounded amount of error to each travel time.

9 Conclusion

We studied admissible strategies in non-zero sum multiplayer timed games. As we showed that admissible strategies do not always exist in the continuous semantics, so we concentrated here on the discrete-time setting. By a reduction to finite concurrent games, we showed the existence of admissible strategies, and gave a characterization of admissible strategies. We gave algorithms to compute the set of admissible outcomes using zone federations, yielding algorithms for model checking under admissibility and assume-admissible synthesis. As future work, we would like to study these symbolic algorithms without the assumption of bounded clocks, thus, using extrapolation operators. We will also implement a prototype tool to test the feasability of our methods.

References

1. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. E. Asarin and O. Maler. As soon as possible: Time optimal control for timed automata. In *Hybrid Systems: Computation and Control, Second International Workshop, HSCC'99, Berg en Dal, The Netherlands, March 29-31, 1999, Proceedings*, volume 1569 of *Lecture Notes in Computer Science*, pages 19–30. Springer, 1999.

3. C. Baier and J. Katoen. *Principles of model checking.* MIT Press, 2008.
4. N. Basset, G. Geeraerts, J. Raskin, and O. Sankur. Admissibility in concurrent games. *CoRR*, abs/1702.06439, 2017.
5. G. Behrmann, A. Cougnard, A. David, E. Fleury, K. G. Larsen, and D. Lime. Uppaal-tiga: Time for playing games! In *Computer Aided Verification, 19th International Conference, CAV 2007, Berlin, Germany, July 3-7, 2007, Proceedings*, volume 4590 of *Lecture Notes in Computer Science*, pages 121–125. Springer, 2007.
6. J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. In J. Desel, W. Reisig, and G. Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 2098 of *Lecture Notes in Computer Science*, pages 87–124. Springer-Verlag, 2004.
7. D. Berwanger. Admissibility in infinite games. In *Proc. of STACS*, LNCS 4393, pages 188–199. Springer, 2007.
8. P. Bouyer, R. Brenguier, and N. Markey. Computing equilibria in two-player timed games *via* turn-based finite games. In *Formal Modeling and Analysis of Timed Systems - 8th International Conference, FORMATS 2010, Klosterneuburg, Austria, September 8-10, 2010. Proceedings*, volume 6246 of *Lecture Notes in Computer Science*, pages 62–76. Springer, 2010.
9. P. Bouyer, R. Brenguier, and N. Markey. Nash equilibria for reachability objectives in multi-player timed games. In *CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings*, volume 6269 of *Lecture Notes in Computer Science*, pages 192–206. Springer, 2010.
10. R. Brenguier, L. Clemente, P. Hunter, G. A. Pérez, M. Randour, J. Raskin, O. Sankur, and M. Sassolas. Non-zero sum games for reactive synthesis. In *Language and Automata Theory and Applications - 10th International Conference, LATA 2016, Prague, Czech Republic, March 14-18, 2016, Proceedings*, volume 9618 of *Lecture Notes in Computer Science*, pages 3–23. Springer, 2016.
11. R. Brenguier, G. A. Pérez, J. Raskin, and O. Sankur. Admissibility in quantitative graph games. In *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2016, December 13-15, 2016, Chennai, India*, volume 65 of *LIPIcs*, pages 42:1–42:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.
12. R. Brenguier, J.-F. Raskin, and O. Sankur. Assume-admissible synthesis. In *Proc. of CONCUR*, LIPIcs 42, pages 100–113. Schloss Dagstuhl–LZI, 2015.
13. R. Brenguier, J.-F. Raskin, and M. Sassolas. The complexity of admissibility in omega-regular games. In *Proc. of CSL-LICS*, pages 23:1–23:10. ACM, 2014.
14. T. Brihaye, V. Bruyère, N. Meunier, and J.-F. Raskin. Weak Subgame Perfect Equilibria and their Application to Quantitative Reachability. In S. Kreutzer, editor, *24th EACSL Annual Conference on Computer Science Logic (CSL 2015)*, volume 41 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 504–518, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
15. T. Brihaye, V. Bruyère, and J. Raskin. On optimal timed strategies. In *Formal Modeling and Analysis of Timed Systems, Third International Conference, FORMATS 2005, Uppsala, Sweden, September 26-28, 2005, Proceedings*, volume 3829 of *Lecture Notes in Computer Science*, pages 49–64. Springer, 2005.
16. T. Brihaye, T. A. Henzinger, V. S. Prabhu, and J. Raskin. Minimum-time reachability in timed games. In *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, volume 4596 of *Lecture Notes in Computer Science*, pages 825–837. Springer, 2007.
17. F. Cassez, A. David, E. Fleury, K. G. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *CONCUR 2005 - Concurrency Theory, 16th International Conference, CONCUR 2005, San Francisco, CA, USA, August 23-26, 2005, Proceedings*, volume 3653 of *Lecture Notes in Computer Science*, pages 66–80. Springer, 2005.
18. K. Chatterjee and T. A. Henzinger. Assume-guarantee synthesis. In *Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 4424, pages 261–275. Springer, 2007.
19. K. Chatterjee, T. A. Henzinger, and M. Jurdziński. Games with secure equilibria. *Theoretical Computer Science*, 365(1):67–82, 2006.

20. K. Chatterjee, T. A. Henzinger, and N. Piterman. Strategy logic. *Information and Computation*, 208(6):677–693, 2010.
21. R. Condurache, E. Filiot, R. Gentilini, and J. Raskin. The complexity of rational synthesis. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 121:1–121:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.
22. D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. In J. Sifakis, editor, *Proceedings of the International Workshop on Automatic Verification Methods for Finite State Systems (AVMFSS'89)*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer, 1990.
23. M. Faella. Admissible strategies in infinite games over graphs. In *Proc. of MFCS*, LNCS 5734, pages 307–318. Springer, 2009.
24. D. Fisman, O. Kupferman, and Y. Lustig. Rational synthesis. In *Proc. of TACAS*, LNCS 6015, pages 190–204. Springer, 2010.
25. P. G. Jensen, K. G. Larsen, and J. Srba. Real-time strategy synthesis for timed-arc petri net games via discretization. In D. Bošnački and A. Wijs, editors, *Model Checking Software: 23rd International Symposium, SPIN 2016, Co-located with ETAPS 2016, Eindhoven, The Netherlands, April 7-8, 2016, Proceedings*, pages 129–146, Cham, 2016. Springer International Publishing.
26. O. Kupferman, G. Perelli, and M. Y. Vardi. Synthesis with rational environments. In *Proc. of EUMAS*, LNCS 8953, pages 219–235. Springer, 2014.
27. O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems (an extended abstract). In *STACS*, pages 229–242, 1995.
28. F. Mogavero, A. Murano, and M. Y. Vardi. Reasoning about strategies. In *Proc. of FSTTCS*, LIPIcs 8, pages 133–144. Schloss Dagstuhl - LZI, 2010.
29. M. Ummels. Rational behaviour and strategy construction in infinite multiplayer games. In *FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science, 26th International Conference, Kolkata, India, December 13-15, 2006, Proceedings*, volume 4337 of *Lecture Notes in Computer Science*, pages 212–223. Springer, 2006.

A Proofs

Proof (of Lemma 2).

- Case $[\![\texttt{closed}(Z)]\!]_{\mathsf{d}} = [\![Z]\!]_{\mathsf{d}}$. The direction $[\![\texttt{closed}(Z)]\!]_{\mathsf{d}} \subseteq [\![Z]\!]_{\mathsf{d}}$ is clear since the operator `closed` strengthens the constraints. We show the other direction. Let $\nu \in [\![Z]\!]_{\mathsf{d}}$. We show that ν satisfies all constraints of $\texttt{closed}(Z)$. It suffices to show that this is the case before the application of canonization (as we will prove that *all* constraints hold). By hypothesis, we have, for all $x, y \in X_0$, $\nu(x) - \nu(y) \prec^Z_{x,y} Z_{x,y}$. If $\prec^Z_{x,y} = \leq$, then $Z_{x,y} = \texttt{closed}(Z)_{x,y}$, so ν satisfies this constraint. If $\prec^Z_{x,y} = <$, then since $\nu \in \mathbb{N}^X$, we have $\nu(x) - \nu(y) \leq Z_{x,y} - 1 = \texttt{closed}(Z)_{x,y}$. Thus, $\nu \in [\![Z]\!]$, and since ν is discrete, $\nu \in [\![Z]\!]_{\mathsf{d}}$.
- Case $\overline{[\![Z]\!]_{\mathsf{d}}}^{\mathsf{d}} = [\![\texttt{closed}(\overline{Z})]\!]_{\mathsf{d}}$. By the previous case, the RHS is equal to $[\![\overline{Z}]\!]_{\mathsf{d}}$. Let $\nu \in \overline{[\![Z]\!]_{\mathsf{d}}}^{\mathsf{d}}$. In particular, $\nu \notin [\![Z]\!]$, which means $\nu \in [\![\overline{Z}]\!]$. But since $\nu \in \mathbb{N}^X$, $\nu \in [\![\overline{Z}]\!]_{\mathsf{d}}$.

  Conversely, if $\nu \in [\![\overline{Z}]\!]_{\mathsf{d}} \subseteq [\![\overline{Z}]\!]$, then $\nu \in \overline{[\![Z]\!]}$. Since ν is discrete, $\nu \in \overline{[\![Z]\!]}^{\mathsf{d}}$, and since complement is nondecreasing, $\overline{[\![Z]\!]}^{\mathsf{d}} \subseteq \overline{[\![Z]\!]_{\mathsf{d}}}^{\mathsf{d}}$.
- Case $[\![Z]\!]_{\mathsf{d}} \cap [\![Z']\!]_{\mathsf{d}} = [\![Z \cap Z']\!]_{\mathsf{d}}$. For any ν, we have $\nu \in [\![Z]\!]_{\mathsf{d}} \cap [\![Z']\!]_{\mathsf{d}}$ if and only if $\nu \in \mathbb{N}^X$ and for all $x, y \in X_0$, $\nu(x) - \nu(y) \prec^Z_{x,y} Z_{x,y}$ and $\nu(x) - \nu(y) \prec^{Z'}_{x,y} Z'_{x,y}$. This is equivalent to the condition $\nu \in \mathbb{N}^X$ and $\nu \in [\![Z \cap Z']\!]$, that is, $\nu \in [\![Z \cap Z']\!]_{\mathsf{d}}$.
- Case $\texttt{Free}^{\mathsf{d}}_R([\![Z]\!]_{\mathsf{d}}) = [\![\texttt{Free}_R(Z)]\!]_{\mathsf{d}}$. The direction $\subseteq$ follows by definition. Let $\nu \in [\![\texttt{Free}_R(Z)]\!]_{\mathsf{d}}$, and let $\nu' \in [\![Z]\!]$ such that $\nu[R \leftarrow 0] = \nu'$. It follows that $\nu' \in [\![Z]\!]_{\mathsf{d}}$. So $\nu \in \texttt{Free}^{\mathsf{d}}_R([\![Z]\!]_{\mathsf{d}})$.

– Case $[\![Z]\!]_{\mathtt{d}}[R \leftarrow 0] = [\![Z[R \leftarrow 0]]\!]_{\mathtt{d}}$. Consider $\nu[\![Z]\!]_{\mathtt{d}}[R \leftarrow 0]$, and let $\nu' \in [\![Z]\!]_{\mathtt{d}}$ such that $\nu'[R \leftarrow 0] = \nu$. Since $\nu' \in [\![Z]\!]$, we have $\nu \in [\![Z[R \leftarrow 0]]\!]$, and since ν is discrete, $\nu \in [\![Z[R \leftarrow 0]]\!]_{\mathtt{d}}$.
Consider now $\nu \in [\![Z[R \leftarrow 0]]\!]_{\mathtt{d}}$, and let $\nu' \in [\![Z]\!]$ such that $\nu'[R \leftarrow 0] = \nu$. Let $\nu''(x)$ be defined as follows. We have $\nu''(x) = \nu'(x)$ for all $x \in X$ such that $\nu'(x) \in \mathbb{N}$. For any other x, let $\nu''(x) = \lfloor \nu'(x) \rfloor$. Notice that for all $x \notin R$, $\nu'(x) = \nu''(x)$ since $\nu(x) = \nu'(x)$ and $\nu \in \mathbb{N}^X$. We show that $\nu'' \in [\![Z]\!]$. Let us write $\nu'(x) - \nu'(y) = \lfloor \nu'(x) \rfloor + \epsilon_1 - \lfloor \nu'(y) \rfloor - \epsilon_2$ where $\epsilon_1, \epsilon_2 \in [0, 1)$. So since $\nu'(x) - \nu'(y) \leq Z_{x,y}$, we have $\lfloor \nu'(x) \rfloor - \lfloor \nu'(y) \rfloor \leq Z_{x,y} + \epsilon$ where $\epsilon = \epsilon_2 - \epsilon_1 \in (-1, 1)$. Since $\epsilon < 1$ and $Z_{x,y} \in \mathbb{N}$, this means that $\lfloor \nu'(x) \rfloor - \lfloor \nu'(y) \rfloor \leq Z_{x,y}$; hence $\nu''(x) - \nu''(y) \leq Z_{x,y}$. It follows that $\nu'' \in [\![Z]\!]_{\mathtt{d}}$, and $\nu''[R \leftarrow 0] = \nu$.
– Case $[\![Z]\!]_{\mathtt{d}}\uparrow^{\mathtt{d}} = [\![Z\uparrow]\!]_{\mathtt{d}}$. The inclusion $\subseteq$ follows by definition. Let $\nu' \in [\![Z\uparrow]\!]_{\mathtt{d}}$, and let $\nu \in [\![Z]\!]$ such that $\nu' = \nu + d$ for some $d \geq 0$. We can assume that ν and d are discrete. In fact, assume otherwise, and write $\nu'' = \nu' - \lfloor d \rfloor = \lceil \nu' - d \rceil$, so that $\nu = \nu'' + \epsilon$ with $\epsilon \in (0, 1)$. Now, both ν and ν'' satisfy the same diagonal constraints since $\nu(x) - \nu(y) = \nu''(x) - \nu''(y)$. For each $x \in X$, we have $\nu''(x) - \epsilon = \nu(x) \leq Z_{x,0}$, that is, $\nu''(x) \leq Z_{x,0} + \epsilon$. Since $\nu''(x)$ and $Z_{x,0}$ are integers, and $\epsilon \in (0, 1)$, we must have $\nu''(x) \leq Z_{x,0}$. Therefore, $\nu'' \in [\![Z]\!]_{\mathtt{d}}$, and $\nu'' + \lfloor d \rfloor = \nu' \in [\![Z]\!]_{\mathtt{d}}\uparrow^{\mathtt{d}}$
– Case $[\![Z]\!]_{\mathtt{d}}\downarrow^{\mathtt{d}} = [\![Z\downarrow]\!]_{\mathtt{d}}$ is symmetric to the previous one.

Proof (of Lemma 3). We first show $\mathtt{Win}_i \supseteq \nu Z.\overline{B_i}^{\mathsf{d}} \cap S \cap \pi_i^{\mathsf{d}}(Z)$. Let $Z_0 = \overline{B_i}^{\mathsf{d}} \cap S$, and $Z_j = \overline{B_i}^{\mathsf{d}} \cap S \cap \pi_i^{\mathsf{d}}(Z_{j-1})$ for all $j \geq 1$, which are the iterates of the fixpoint. We show that from each Z_j, Player i has a strategy to avoid B_i for at least j discrete transitions. This is clear for Z_0. Consider state $q \in Z_j \subseteq \overline{B_i}^{\mathsf{d}} \cap S$ so that q does not immediately violate Player-i's objective. Since $q \in \pi_i^{\mathsf{d}}(Z_{j-1})$, there exists $d \in \mathbb{N}$ such that $q + d \in \mathtt{Pred}_i(Z_{j-1})$ and $q + [0, d] \cap \mathtt{Pred}_{-i}(\overline{Z_{j-1}}^{\mathsf{d}}) = \emptyset$. We let Player i wait for $d - 1$ times, and from $q + d$, pick an edge that goes to Z_{j-1}. Hence, either this edge is taken from $q + d$, or some other player's edge is taken from some state in $q + [0, d]$. By definition of π_i, in both cases, the next state is in Z_{j-1}, and we can conclude by induction.

To show the converse, consider state $q \in S \setminus Z_j$ for some j. We show that under any strategy of Player i, there is an outcome which, from q, reaches B_i in at most j steps; hence $q \notin \mathtt{Win}_i$. We prove this by induction on j. The property is trivial if $j = 0$ since the objective is immediately violated. Assume $j \geq 1$. We have by hypothesis $q \notin \mathtt{TPred}^{\mathsf{d}}(\mathtt{Pred}_i(Z_{j-1}), \mathtt{Pred}_{-i}(\overline{Z_{j-1}}^{\mathsf{d}}))$, for all delays $d \in \mathbb{N}$ Player i might wait for (that is, $q + d \in S$), either $\mathtt{Pred}_i(q + d) \cap Z_{j-1} = \emptyset$ or $q + [0, d] \cap \mathtt{Pred}_{-i}(\overline{Z_{j-1}}^{\mathsf{d}}) \neq \emptyset$. Thus if a strategy, at q, waits for d steps and chooses an edge e, either this edge does not enter Z_{j-1} or $q + [0, d]$ contains a state where another player can move the play outside of Z_{j-1}. So some outcome under Player-i's strategy is outside Z_{j-1} at the next step. We conclude by induction.

Proof (of Lemma 4). The proof of $\mathtt{TPred}^{\mathsf{d}}(G, B) = \bigcup_k \bigcap_j \mathtt{TPred}^{\mathsf{d}}(G_k, B_j)$ is identical to the counterpart in [17, Lemma 4].

We show $\mathtt{TPred}^{\mathsf{d}}(G_k, B_j) = [\![\mathtt{Inv} \cap \left((G_k\downarrow \cap \overline{B_j\downarrow}^{\mathsf{d}}) \cup \left(G_k \cap (B_j\downarrow) \cap \overline{B_j}^{\mathsf{d}}\right)\downarrow\right)]\!]_{\mathtt{d}}$.

We start with $\mathtt{TPred}^{\mathsf{d}}(G_k, B_j) \subseteq [\![\mathtt{Inv} \cap \left((G_k\downarrow \cap \overline{B_j\downarrow}^{\mathsf{d}}) \cup \left(G_k \cap (B_j\downarrow) \cap \overline{B_j}^{\mathsf{d}}\right)\downarrow\right)]\!]_{\mathtt{d}}$. Let $q \in \mathtt{TPred}^{\mathsf{d}}(G_k, B_j)$. By definition of $\mathtt{TPred}^{\mathsf{d}}$, $q \in \mathtt{Inv}$. Let $d \in \mathbb{N}$ such that $q + d \in G_k$ while $q + [0, d] \cap B_j = \emptyset$. We thus have $q \in G_k\downarrow$. Now, if $q \notin B_j\downarrow$, then also $q \in \overline{B_j\downarrow}^{\mathsf{d}}$ since q is discrete. Hence, q belongs to the first term of the right hand side. Otherwise, we have $q \in B_j\downarrow$. Let d' be such that $q + d' \in B_j$. By hypothesis, we have $q + [0, d] \cap B_j = \emptyset$, so we must have $d < d'$. So $q + d \in \overline{B_j}^{\mathsf{d}}$, and since $(q + d) + d' \in B_j$, we have $q + d \in G_k \cap B_j\downarrow$. Thus, q belongs to the second term.

We now show $[\![\mathtt{Inv} \cap \left((G_k{\downarrow} \cap \overline{B_j{\downarrow}}^{\mathsf{d}}) \cup \left(G_k \cap (B_j{\downarrow}) \cap \overline{B_j}^{\mathsf{d}}\right){\downarrow}\right)]\!]_{\mathsf{d}} \subseteq \mathtt{TPred}^{\mathsf{d}}(G_k, B_j)$. Consider state q in the left hand side. If $q \in \mathtt{Inv} \cap (G_k{\downarrow} \cap \overline{B_j{\downarrow}}^{\mathsf{d}})$, then there exists $d \in \mathbb{N}$ such that $q+d \in G_k$ and $q + [0,d] \cap B_j = \emptyset$ since $q \notin B_j{\downarrow}$. Consider now $q \in \mathtt{Inv} \cap (G_k \cap (B_j{\downarrow}) \cap \overline{B_j}^{\mathsf{d}}){\downarrow}$. Let $d \in \mathbb{N}$ such that $q + d \in G_k \cap (B_j{\downarrow}) \cap \overline{B_j}^{\mathsf{d}}$. We claim that $q + [0,d] \cap B_j = \emptyset$. Assume otherwise, and let $d' \in [0,d]$ with $q + d' \in B_j$. Since $q + d \in \overline{B_j}^{\mathsf{d}} \cap B_j{\downarrow}$, there must exist $d'' \in \mathbb{N}$ such that $q + d + d'' \in B_j$. By convexity, $[q + d', q + d + d''] \subseteq B_j$, and we have $q + d \in B_j$ which is a contradiction.

Proof (of Lemma 6).

- Consider a reduced DBM D and let D' denote the DBM obtained by $\mathtt{Shift}_{-1,-1}(D)$ before reduction. Let us show that $\nu \in [\![D']\!]_{\mathsf{d}}$ if, and only if $\nu + 1 \in [\![D]\!]_{\mathsf{d}}$. Consider $\nu \in [\![D']\!]_{\mathsf{d}}$. For diagonal constraints $x, y \in X$, if $\nu(x) < M$, then $(\nu +_M 1)(x) - (\nu +_M)(y) = \nu(x) - \nu(y) \leq D'_{x,y} = D_{x,y}$. Otherwise,
  For diagonal constraints $x, y \in X$, we have $(\nu+1)(x) - (\nu+1)(y) = \nu(x) - \nu(y) \prec^D_{x,y} D_{x,y} = D'_{x,y}$ which establishes the equivalence for these constraints. Consider $x \in X$. We clearly have $(\nu+1)(x) \prec^D_{x,0} D_{x,0}$ if, and only if $\nu(x) \prec^D_{x,0} D_{x,0} - 1$. Similarly, $-(\nu+1)(x) \prec^D_{0,x} D_{0,x}$ if, and only if $-\nu(x) \prec^D_{0,x} D_{0,x} + 1$.
- Consider a reduced DBM D and let D' denote the DBM obtained by $\mathtt{Shift}_{0,-1}(D)$ before reduction. We show that $\nu \in [\![D']\!]_{\mathsf{d}}$ iff $\nu \in [\![D]\!]_{\mathsf{d}}$ and $\nu + 1 \in [\![D]\!]_{\mathsf{d}}$. The equivalence is obvious for diagonal constraints (see previous case). Fix $x \in X$. We have that $\nu(x) \prec_{x,0} D_{x,0} - 1$ implies $\nu(x) \prec_{x,0} D_{x,0}$ and $\nu(x) + 1 \prec_{x,0} D_{x,0}$. Moreover, if $\nu(x) + 1 \prec_{x,0} D_{x,0}$, then $\nu(x) \prec_{x,0} D_{x,0} - 1$, which establishes the equivalence for constraints of type $(x, 0)$. Last, we have that $-\nu(x) \prec_{0,x} D_{0,x}$ implies $-\nu(x) - 1 \prec_{0,x} D_{0,x}$. It follows that $-(\nu+1)(x) \prec_{0,x} D_{0,x}$ and $-\nu(x) \prec_{0,x} D_{0,x}$ iff $-\nu(x) \prec_{0,x} D'_{0,x} = D_{0,x}$.
- Consider a reduced DBM D and let D' denote the DBM obtained by $\mathtt{Shift}_{-1,0}(D)$ before reduction. We show that $\nu \in [\![D']\!]_{\mathsf{d}}$ iff $\nu \in [\![D]\!]_{\mathsf{d}}$ or $\nu + 1 \in [\![D]\!]_{\mathsf{d}}$. The proof is again easy for diagonal constraints. Let us fix $x \in X$. We have $\nu(x) \prec^D_{x,0} D_{x,0}$ iff $\nu(x) \prec^D_{x,0} D_{x,0}$ or $\nu(x) + 1 \prec^D_{x,0} D_{x,0}$. Furthermore, $-\nu(x) \prec^D_{0,x} D'_{0,x} = D_{0,x} + 1$ iff $-(\nu+1)(x) \prec^D_{0,x} D_{0,x}$. This proves the required equivalence.
- Consider a reduced DBM D and let D' denote the DBM obtained by $\mathtt{Shift}_{0,1}(D)$ before reduction. We show that $\nu \in [\![D']\!]_{\mathsf{d}}$ iff $\nu \in [\![D]\!]_{\mathsf{d}}$ or $\nu - 1 \in [\![D]\!]_{\mathsf{d}}$. The proof is again easy for diagonal constraints. Let us fix $x \in X$. We have $\nu(x) \prec^D_{x,0} D'_{x,0} = D_{x,0} + 1$ iff $\nu(x) \prec^D_{x,0} D_{x,0}$ or $\nu(x) - 1 \prec^D_{x,0} D_{x,0}$, Furthermore, $-\nu(x) \prec^D_{0,x} D'_{0,x} = D_{0,x}$ iff $-\nu(x) \prec^D_{0,x} D_{0,x}$ or $-(\nu - 1)(x) \prec^D_{0,x} D_{0,x}$. This proves the required equivalence.

Proof (of Lemma 8). We show the equivalence of the set defined by $[\![A^i]\!]_{\mathsf{d}}$, and that of Theorem 3 for $\bot$.

Notice that, by Theorem 3, any move is admissible at $\mathtt{Lose}_i$; this matches the first term in A^i. Second, waiting is admissible from $\mathtt{Win}_i$ if and only if the current state and the immediate time successor both belong to $\mathtt{Win}_i$; this is exactly captured by the term $\mathtt{Shift}_{0,-1}(\mathtt{Win}_i)$. Consider now states of $\mathtt{Maybe}_i \cap \overline{\mathtt{Pred}_i(\mathtt{Win}_i)}$. By Theorem 3, no state of $\mathtt{Maybe}_i$ outside this set is locally admissible. Now it remains to show that B^i describes the states satisfying item 1, and C^i item 2.

Consider B^i. The set $\mathtt{Shift}_{-1,-1}(\mathtt{Win}_i \cup \mathtt{Maybe}_i)$ describes exactly those states q such that $q +_M 1 \in \mathtt{Maybe}_i \cup \mathtt{Win}_i$. Notice that no state q satisfies $q +_M 1 = q$ in our setting due to the hypothesis on invariants bounding all clocks.

We now consider C^i. It describes the set of states at which, for any edge $e \in \Delta_i$ whose guard is satisfied (g_e), and which leads to $\mathtt{Maybe}_i$ ($\mathtt{Pred}_e(\mathtt{Maybe}_i)$), there is $e' \in \Delta_{-i}$ whose guard is

satisfied, which does not lead to $\texttt{Lose}_i$, and has a different successor than e. Thus, this captures item 2.